

MATLAB 仿真与应用系列丛书

MATLAB 程序设计 与典型应用

张德丰 编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

MATLAB 语言是现今在工程研究领域应用范围很广的一门计算机语言。本书讲解了 MATLAB 语言运算、程序设计、图形表示,同时对 MATLAB 语言在工程中的经典应用进行了详细介绍。

本书共分 11 章,包括 MATLAB 简介、MATLAB 数值计算及应用、符号运算及应用、MATLAB 程序设计技术、MATLAB 绘图功能、MATLAB 在模糊控制系统中的应用、MATLAB 在人工神经网络中的应用、MATLAB 在自动控制中的应用、MATLAB 在数字信号中的应用、MATLAB 外部程序接口应用、MATLAB 在其他领域的应用等内容。

本书结构清晰、内容丰富、论述翔实,适合学习 MATLAB 的本科生、研究生阅读,也可作为广大科研工作人员的参考用书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

MATLAB 程序设计与典型应用 / 张德丰编著.—北京:电子工业出版社,2009.6
(MATLAB 仿真与应用系列丛书)

ISBN 978-7-121-08874-2

I. M... II. 张... III. 计算机辅助计算—软件包, MATLAB—程序设计 IV. TP391.75

中国版本图书馆 CIP 数据核字(2009)第 077720 号

策划编辑:陈韦凯

责任编辑:李雪梅

印 刷:

装 订:

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1092 1/16 印张:22.25 字数:570 千字

印 次:2009 年 6 月第 1 次印刷

印 数:4 000 册 定价:39.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn,盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

前 言

MATLAB 是 Matrix Laboratory (“矩阵实验室”)的缩写,是由美国 MathWorks 公司开发的集数值计算、符号运算和图形可视化三大基本功能于一体的,功能强大、操作简单的语言,是国际公认的优秀数学应用软件之一。“从工程师和科学家的角度来看, MATLAB 有许多的优点,是它的同类产品中最好的软件”。

随着 MATLAB 在各个工程领域应用的日益广泛,专家学者们相继推出了控制系统工具箱 (Control Systems Toolbox)、模糊逻辑工具箱 (Fuzzy Logic Toolbox)、虚拟现实工具箱 (Virtual Reality Toolbox)、数字信号处理模块库 (DSP Blockset)、神经网络模块库 (Neural NetWork Blockset) 和航天器控制模块库 (Aerospace Blockset) 等简单实用的工具箱和模块库,这些工具箱给各个领域的研究和工程应用提供了强有力的工具,而且这些工具箱还在不断增加。目前推出的 MATLAB R2008 不但扩展和完善了某些工具箱的功能,还添加了新的工具箱。因此,借助于日渐完善的 MATLAB 软件,各个领域的研究人员可以直观、方便地进行分析、计算和设计工作,大大节省了时间,提高了工作效率。此外,在 MATLAB 中,所有的分析工具都可以立即获得,因此可以很方便地看到运行结果、分析这些结果,并且使结果可视化。

MATLAB 具有其他高级语言难以比拟的一些优点,编写简单,编程效率高,易学易懂,因此 MATLAB 语言也被通俗地称为演算纸式的科学算法语言。在控制、通信、信号处理及科学计算等领域中, MATLAB 都被广泛地应用,已经被认可为能够有效提高工作效率、改善设计手段的工具软件,掌握了 MATLAB 就好比掌握了开启这些专业领域大门的钥匙。

本书是在充分体现应用型教育特点,提高学生分析问题及解决问题能力的基础上编写的,具有以下特点:

(1) 精选内容,条理清晰。全书以基础知识为主,科学新成果与发展新动向相结合,系统地展开介绍 MATLAB 的编程基础与典型应用。

(2) 重点突出,目的明确。立足基本理论、面向应用技术,以必须、够用为尺度,以掌握概念、强化应用为重点,加强理论知识和实际应用的统一。

(3) 注重实用,强化实践。以 MATLAB 为编程工具,通过大量典型实例的分析和实践,使读者较快地认识到 MATLAB 软件可以方便、快捷地编程,以及 MATLAB 在各个领域中的典型应用。

全书共分 11 章。第 1 章介绍了 MATLAB 基础知识,包括 MATLAB 启动与安装、MATLAB 的开发环境、MATLAB 帮助系统等内容;第 2 章介绍了 MATLAB 数值计算及应用, MATLAB 的数组、矩阵运算、MATLAB 多项式及其运算等内容;第 3 章介绍了符号运算及应用, MATLAB 符号微积分运算、复变函数运算的 MATLAB 实现等内容;第 4 章介绍了 MATLAB 程序设计技术,包括 MATLAB 的控制语句、M 文件编程等内容;第 5 章介绍了 MATLAB 绘图功能,包括二维图形绘制、三维图形绘制、图形颜色的应用等内容;第 6 章介绍了 MATLAB 在模糊控制系统中的应用,包括模糊系统的 MATLAB 实现、MATLAB 模糊逻辑工具箱命令函数等内容;第 7 章介绍了 MATLAB 在人工神经网络中的应用,包括感知器、BP 网络、径向基网络等内容;第 8 章介绍了 MATLAB 在自动控制中的应用,包括控制系统模型、系统校正等内容;第 9 章

介绍了 MATLAB 在数字信号中的应用，包括数字信号知识、离散时间傅里叶变换、数字滤波器的分析与实现等内容；第 10 章介绍了 MATLAB 外部程序接口应用，包括 MATLAB 数据接口、MATLAB 编译器的配置等内容；第 11 章介绍了 MATLAB 的其他领域应用，包括 MATLAB 在电路中的应用、MATLAB 在优化设计中的应用等内容。

为方便读者阅读，书中部分字母、符号等采用正体。

由于时间仓促，加之作者水平有限，所以错误和疏漏之处在所难免。在此，诚恳地期望得到各领域专家和广大读者的批评指正。

作 者
2009 年 3 月

目 录

第 1 章	MATLAB 简介	1
1.1	MATLAB 概述	1
1.2	MATLAB 环境	1
1.2.1	MATLAB 安装	1
1.2.2	MATLAB 启动与退出	4
1.3	MATLAB 的开发环境	5
1.3.1	工作界面	5
1.3.2	命令窗口	6
1.3.3	当前目录浏览器窗口	8
1.3.4	工作空间浏览器窗口	9
1.3.5	历史命令窗口	10
1.3.6	数组编辑器窗口	11
1.4	MATLAB 帮助系统	11
1.4.1	帮助命令	11
1.4.2	帮助窗口	12
1.4.3	演示系统	13
第 2 章	MATLAB 数值计算及应用	15
2.1	MATLAB 的数值计算基础	15
2.1.1	数据类型	15
2.1.2	常量和变量	18
2.1.3	数值计算应用的示例	19
2.2	MATLAB 的数组、矩阵运算	20
2.2.1	数组与矩阵的概念	20
2.2.2	数组或矩阵元素的标识	21
2.2.3	数组与矩阵的输入	23
2.2.4	数组与矩阵的算术运算	25
2.2.5	向量及其运算	32
2.2.6	矩阵的特殊运算	35
2.2.7	数组的运算	44
2.2.8	字符串	47
2.3	MATLAB 多项式及其运算	47
2.3.1	多项式求值	48
2.3.2	多项式求根	48
2.3.3	部分分式展开	49
2.3.4	多项式乘除	50
2.3.5	多项式的微积分	50

2.4	插值与拟合	51
2.4.1	一维插值问题.....	51
2.4.2	二维插值问题.....	52
2.4.3	曲线拟合.....	56
2.5	线性方程组求解.....	58
2.5.1	方程组解法.....	58
2.5.2	求线性方程组的通解.....	65
2.6	非线性方程与最优化问题.....	67
2.6.1	非线性方程数值求解.....	67
2.6.2	无约束最优化问题求解.....	69
2.6.3	有约束最优化问题求解.....	71
第3章	符号运算及应用.....	73
3.1	MATLAB 符号运算基础	73
3.1.1	符号表达式.....	73
3.1.2	符号表达式的操作及代数运算.....	75
3.2	MATLAB 符号微积分运算	82
3.2.1	符号极限运算.....	82
3.2.2	符号函数微分运算.....	82
3.2.3	符号函数积分运算.....	83
3.2.4	符号求和函数与级数展开函数.....	84
3.3	复变函数运算的 MATLAB 实现	86
3.3.1	复数的概念.....	86
3.3.2	复变量的函数.....	86
3.3.3	复数的生成及其矩阵创建.....	87
3.3.4	复数的几何意义.....	88
3.3.5	MATLAB 在复数代数运算中的实现.....	90
第4章	MATLAB 程序设计技术	93
4.1	MATLAB 的控制语句	93
4.1.1	条件控制.....	93
4.1.2	循环控制.....	96
4.1.3	错误控制.....	98
4.2	M 文件编程.....	99
4.2.1	M 文件的分类介绍	99
4.2.2	函数调用和变量传递.....	101
4.2.3	数据导入与导出.....	107
4.2.4	示例分析.....	111
4.3	函数类型	118
4.3.1	主函数	118
4.3.2	子函数	119
4.3.3	私有函数.....	119



4.3.4	嵌套函数.....	120
4.3.5	重载函数.....	124
4.4	调试程序	124
4.4.1	调试程序介绍.....	124
4.4.2	MATLAB 调试菜单	125
4.4.3	调试命令.....	126
第 5 章	MATLAB 绘图功能.....	129
5.1	二维图形绘制	129
5.1.1	绘制二维曲线的常用函数.....	129
5.1.2	绘制图形的辅助操作.....	132
5.1.3	绘制二维图形的其他函数.....	137
5.2	三维图形绘制	141
5.2.1	绘制三维曲线的常用函数.....	141
5.2.2	三维曲面图绘制.....	142
5.2.3	其他三维图形绘制.....	146
5.2.4	透明度作图.....	147
5.2.5	立体可视化.....	148
5.3	图形颜色映像的应用.....	151
5.4	光照和材质处理.....	153
5.4.1	光照处理.....	153
5.4.2	材质处理.....	154
5.5	图像显示技术	156
5.5.1	图像简介.....	156
5.5.2	图像的读取.....	157
5.5.3	图像的显示.....	159
5.6	动画制作技术	159
第 6 章	MATLAB 在模糊控制系统中的应用.....	161
6.1	模糊系统的 MATLAB 实现	161
6.1.1	模糊集简介.....	161
6.1.2	模糊推理系统与 MATLAB 应用	166
6.1.3	模糊推理系统的 MATLAB 模糊工具箱的图形界面实现法	166
6.2	MATLAB 模糊逻辑工具箱命令函数及示例.....	173
6.3	MATLAB 模糊逻辑工具箱命令函数应用示例.....	193
第 7 章	MATLAB 在人工神经网络中的应用.....	197
7.1	人工神经网络介绍.....	197
7.2	感知器	197
7.2.1	感知器原理.....	198
7.2.2	感知器相关函数.....	199
7.2.3	感知器的 MATLAB 实现	202
7.3	线性神经网络	203



7.3.1	线性神经网络原理.....	204
7.3.2	线性神经网络相关函数.....	204
7.3.3	线性神经网络的 MATLAB 实现	206
7.4	BP 网络	207
7.4.1	BP 网络原理.....	207
7.4.2	BP 网络相关函数.....	208
7.4.3	BP 网络的 MATLAB 实现	217
7.5	径向基网络	218
7.5.1	径向基网络原理.....	218
7.5.2	径向基网络相关函数.....	219
7.5.3	径向基网络应用示例.....	222
7.6	回归网络	223
7.6.1	回归网络相关函数.....	223
7.6.2	回归网络的 MATLAB 实现	224
第 8 章	MATLAB 在自动控制中的应用	227
8.1	控制系统模型	227
8.1.1	控制系统的描述与 LTI 对象.....	227
8.1.2	典型系统的生成.....	228
8.1.3	连续系统与采样系统之间的转换	230
8.2	控制系统的时域分析.....	231
8.2.1	时域分析的一般方法.....	231
8.2.2	常用时域分析函数.....	234
8.2.3	时域分析应用示例.....	237
8.3	根轨迹分析	238
8.3.1	模条件和角条件.....	239
8.3.2	绘制根轨迹的规则.....	239
8.3.3	根轨迹的应用示例.....	240
8.4	控制系统的频域分析.....	243
8.4.1	幅相频率特性.....	243
8.4.2	对数频率特性.....	245
8.4.3	对数幅相特性.....	248
8.5	系统校正	248
8.5.1	串联超前校正.....	248
8.5.2	串联滞后校正.....	251
8.5.3	串联滞后—超前校正.....	252
8.6	极点配置设计方法.....	255
8.6.1	Gura-Bass 算法.....	255
8.6.2	Ackermann 配置算法	256



第9章 MATLAB 在数字信号中的应用	259
9.1 数字信号知识	259
9.1.1 信号产生	259
9.1.2 信号的运算	265
9.1.3 信号的抽取与插值	267
9.2 离散时间傅里叶变换	269
9.2.1 离散时间傅里叶变换定义及计算	269
9.2.2 离散时间傅里叶变换的特性	271
9.3 数字滤波器的分析与实现	273
9.3.1 数字滤波器知识	273
9.3.2 数字滤波器的分析与实现	274
9.4 IIR 数字滤波器的设计法	278
9.4.1 冲激响应不变法	278
9.4.2 双线性变换法	279
9.4.3 IIR 数字滤波器的频率变换设计法	279
9.5 FIR 数字滤波器设计法	283
9.5.1 窗函数设计法	283
9.5.2 频率抽样法	286
9.5.3 MATLAB 的其他相关函数	289
9.6 MATLAB 实现功率谱估计	292
第10章 MATLAB 外部程序接口应用	295
10.1 MATLAB 数据接口	295
10.1.1 通用文件 I/O 操作	295
10.1.2 低级文件 I/O 操作	296
10.1.3 MAT 文件及其应用	300
10.2 MATLAB 编译器的配置	304
10.2.1 MATLAB 编译器的配置	304
10.2.2 编译指令	305
10.3 MATLAB 引擎	306
10.3.1 MATLAB DDE 服务器与引擎库	306
10.3.2 C 语言 MATLAB 引擎	307
10.3.3 Fortran 语言 MATLAB 引擎	310
10.4 Visual C++与 MATLAB 接口	311
10.4.1 Visual C++调用 MATLAB 引擎	312
10.4.2 Visual C++使用数学函数库	312
10.4.3 Visual C++创建 MAT 文件	314
10.4.4 应用 COM 实现 Visual C++与 MATLAB 的接口	316
第11章 MATLAB 在其他领域的应用	323
11.1 MATLAB 在电路中的应用	323
11.1.1 概述	323



11.1.2	MATLAB 在电路中的应用示例.....	323
11.2	MATLAB 在图像处理中的应用.....	326
11.2.1	图像变换.....	327
11.2.2	MATLAB 实现图像的边缘检测.....	328
11.2.3	MATLAB 在遥感中实现图像直方图的匹配.....	330
11.3	MATLAB 在力学及工程结构中的应用.....	333
11.3.1	概述.....	333
11.3.2	MATLAB 在力学及工程结构中的应用示例.....	333
11.4	MATLAB 在优化设计中的应用.....	338
11.4.1	概述.....	338
11.4.2	MATLAB 在优化设计中的应用示例.....	339
参考文献.....		342

第 1 章 MATLAB简介

1.1 MATLAB概述

MATLAB 是一种用于科学技术计算的高性能语言。它将计算、可视化和程序设计集成在一个非常容易使用的环境中，使用我们熟悉的数学符号表示问题与答案。**MATLAB** 的应用范围广泛，包括数学与计算；算法开发；数据采集；建模与模拟；数据分析、研究和可视化；科学和工程图形；应用程序开发，包括图形用户界面的建立。

MATLAB 是一个交互系统，它的基本数据元素是数组，尤其适合解决用矩阵和向量组成数据的科学技术计算问题。

MATLAB 很重要的特点是附加了一个解决专门问题的应用程序大家族，名为工具箱。它对于 **MATLAB** 用户是非常重要的，能让用户学习和应用专门的技术。工具箱是 **MATLAB** 函数的全面集合，扩展了 **MATLAB** 解决特殊类型问题的环境。工具箱可应用的领域包含信号处理、控制系统、神经网络、模糊逻辑、子波、模拟等方面。

MATLAB 这个名字，代表 Matrix Laboratory。

MATLAB 系统由 5 个主要部分组成。

(1) 开发环境：这是一组工具和程序，帮助用户使用 **MATLAB** 功能和文件。许多工具是图形用户界面，包括 **MATLAB** 桌面和命令窗口，命令的历史窗口，编辑器和查错程序，观看帮助信息的浏览器，工作区，文件和搜索路径。

(2) **MATLAB** 的数学函数库：这是一个计算算法的巨大集合，范围从初等函数，如求和、正弦、余弦和复数运算，到更高级的函数，如矩阵求逆、矩阵特征值、贝塞尔函数和快速傅里叶变换。

(3) **MATLAB** 语言：一个高级的矩阵/数组语言，具有控制流语句、函数、数据结构、输入/输出和面向对象的程序设计特点。用这种语言能够快速建立运行快且短小的程序，也能建立大的复杂的应用程序。

(4) 图形：**MATLAB** 有广泛的程序，用于把向量和矩阵显示为图形，以及注释和打印这些图形。它包括高级功能，用于二维和三维数据的形象化、图像处理、动画和演示图形；还包括低级功能，让用户完全定制图形的外观，以及为用户的应用程序建立完全的图形用户界面。

(5) **MATLAB** 应用程序接口 (API)：这是一个程序库，允许用户编写 C 和 Fortran 程序与 **MATLAB** 交互。其中包含的程序，用于从 **MATLAB** 调用程序，调用 **MATLAB** 作为计算引擎，以及读写 MAT 文件。

1.2 MATLAB环境

1.2.1 MATLAB安装

使用 **MATLAB** 前需要安装软件，具体的安装步骤如下所示。

(1) 打开 MATLAB R2008 的安装软件, 启动 setup 文件, 显示如图 1-1 所示的 “Installer Welcome” 窗口, 选择 “Install manually without using the Internet” 选项, 然后单击 “Next” 按钮进入安装过程的下一步。

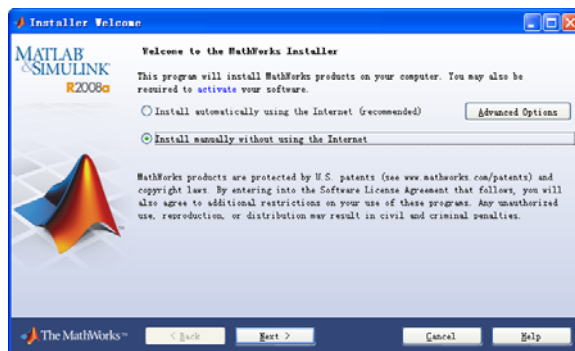


图 1-1 “Installer Welcome” 窗口

(2) 在弹出的如图 1-2 所示的 “License Agreement” 窗口中, 选择 “Yes” 选项, 然后单击 “Next” 按钮, 进入安装下一步。

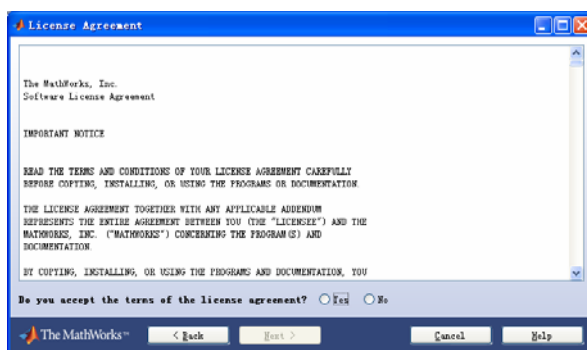


图 1-2 “License Agreement” 窗口

(3) 进入如图 1-3 所示的 “File Installation Key” 窗口, 选中 “I have the File Installation Key for my license” 选项, 在下面的文本框中填入安装密码, 单击 “Next” 按钮, 进入安装下一步。

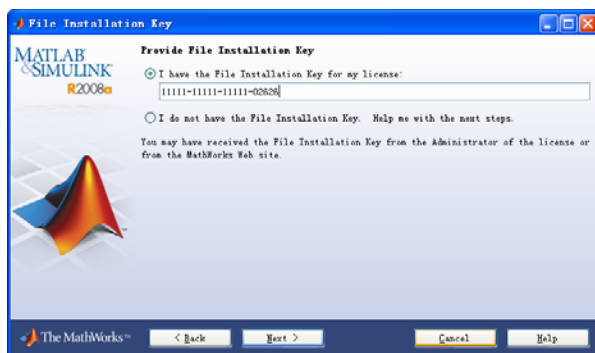


图 1-3 “File Installation Key” 窗口

(4) 在弹出的如图 1-4 所示的 “Installation Type” 窗口中包含 “Typical” 和 “Custom” 两个选项, 选择前者将只安装一般常用的典型组件, 而选择后者, 用户则可以根据自己的需要详

细选择安装的组件。默认安装是“Typical”，一般用户可以选择默认选项，然后单击“Next”按钮进入安装下一步。

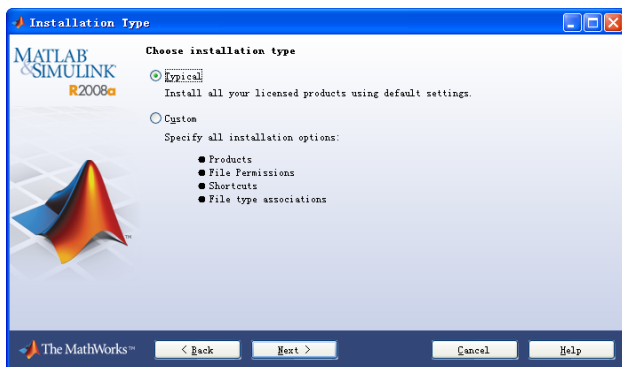


图 1-4 “Installation Type” 窗口

(5) 在弹出的如图 1-5 所示的“Folder Selection”窗口，选择“Browse”按钮，窗口中选择 MATLAB R2008 软件安装的路径，然后单击“Next”按钮进入安装下一步。

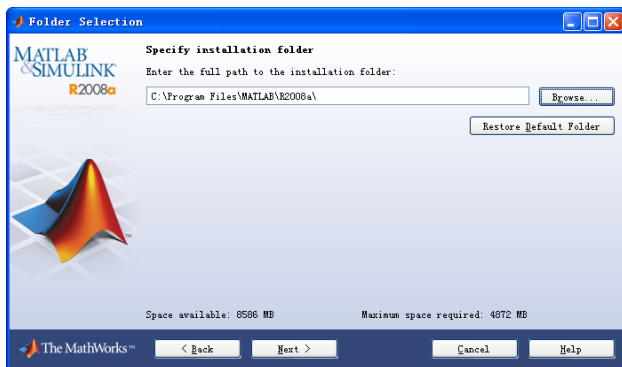


图 1-5 “Folder Selection” 窗口

(6) 在弹出的如图 1-6 所示的“Confirmation”窗口中，单击“Install”按钮，进入安装下一步。

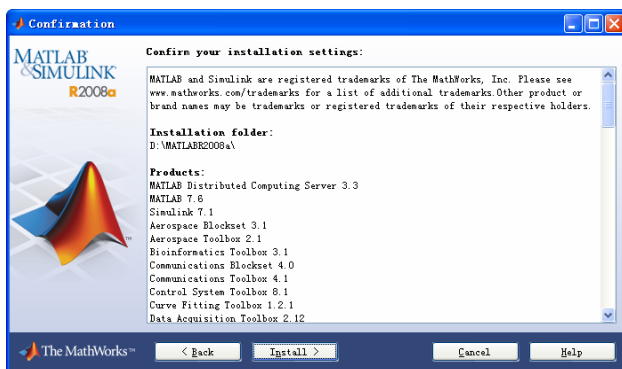


图 1-6 “Confirmation” 窗口

(7) 安装开始进行，如图 1-7 所示。安装完成后如图 1-8 所示，单击“Finish”按钮结束安装。

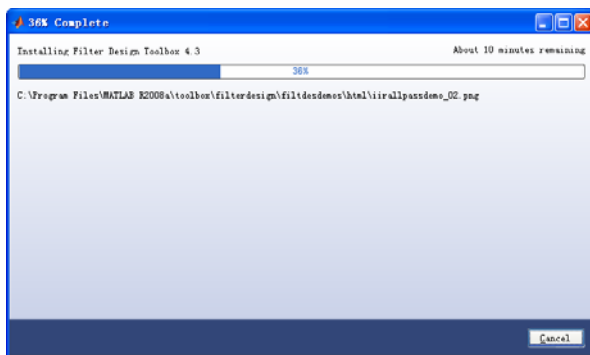


图 1-7 “安装进度表”窗口

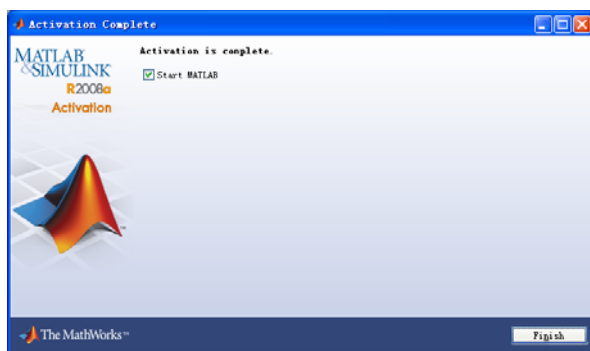


图 1-8 “安装完成”窗口

1.2.2 MATLAB启动与退出

MATLAB R2008 的启动和退出，根据使用系统的不同有着不同的操作。

(1) 启动

在 Windows 和 Macintosh 系统中，程序通常通过双击图标而启动（桌面上的快捷方式或者开始菜单中的程序）。

在 UNIX 系统中，程序是通过在命令行系统提示符后输入如下字符启动的：

```
MATLAB
```

(2) 退出

如果要退出 MATLAB 系统，可以直接在命令窗口中输入“quit”命令，并用回车来退出当前系统。

```
>>quit
```

另外还可以通过单击【File】菜单下的【quit】选项或直接单击 MATLAB 窗口右上角的“关闭”按钮来退出当前系统。

如果想要终止 MATLAB 正在运行的命令或程序，可以同时按下“Ctrl”键和“C”键。MATLAB 将停止正在运行的所有工作，并且在屏幕上给出提示符，等待用户输入。



命令应在提示符“>>”后输入，在本书后面的其他章节中凡是要求输入的命令，均是输入提示符“>>”后的字符。

1.3 MATLAB的开发环境

MATLAB 的默认工作界面如图 1-9 所示。工作界面中包含几个非常重要的工作窗口，如命令窗口、M 文件窗口、起始面板窗口、工作空间窗口、命令历史窗口、当前目录窗口和图形窗口等，下面将对几个常用窗口的功能及使用进行介绍。

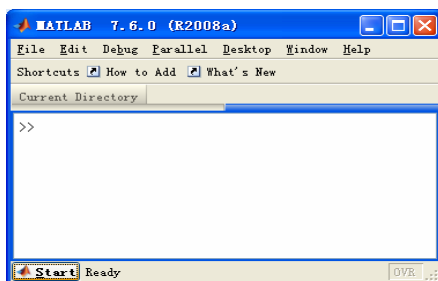


图 1-9 MATLAB 的默认工作界面

1.3.1 工作界面

MATLAB R2008 提供了“File”、“Edit”、“Debug”、“Desktop”、“Window”和“Help”菜单。下面对“File”、“Edit”和“Desktop”菜单进行简单的说明。

(1) File 菜单栏主要负责新建 M 文件、图形窗口、仿真模型和 GUI 设计模型，以及数据导入、路径和属性设置及退出等功能，各命令的具体功能如表 1-1 所示。

表 1-1 File 菜单功能表

下拉菜单		功 能
New	M-file	新建一个 M 文件，打开 M 文件编辑/调试器
	Figure	新建一个图形窗口
	Model	新建一个仿真模型
	GUI	新建一个图形用户设计界面(GUI)
Open		打开已有文件
Close Command History		关闭历史命令窗口
Import Data		导入其他文件的数据
Save Workspace as		使用二进制 MAT 文件保存工作空间的内容
Page Setup		页面设置
Set Path		设置搜索路径等
Preference		设置 MATLAB 工作环境外观和操作的相关属性等参数
Print		打印
Print Selection		打印所选择区域
Exit MATLAB		退出 MATLAB

(2) “Edit”菜单如图 1-10 所示，“Edit”菜单的各菜单项与 Windows 的“Edit”菜单相似。“Paste to Workspace”可以用来打开数据输入向导对话框“Import Wizard”，将剪贴板的数据输入到 MATLAB 工作空间中。

Undo	Ctrl+Z
Redo	
Cut	Ctrl+W
Copy	Alt+W
Paste	Ctrl+Y
Paste to Workspace...	
Select All	
Delete	Ctrl+D
Find and Replace...	
Find Files...	
Clear Command Window	
Clear Command History	
Clear Workspace	

图 1-10 “Edit” 菜单

(3) “Desktop” 菜单主要负责窗口的显示，各菜单命令的具体功能如表 1-2 所示。

表 1-2 Desktop 菜单功能表

下拉菜单	功 能	下拉菜单	功 能
Unlock Command Window	与命令窗口分离	Workspace	打开工作空间窗口
Move Command Window	移动命令窗口	Help	打开帮助窗口
Resize Command Window	重新定义命令窗口大小	Profiler	打开程序性能剖析窗口
Desktop Layout	界面布局 (可选择各种布局方式)	Editor	打开 M 文件编辑器
Command Window	打开命令窗口	Figures	打开图形输出界面
Command History	打开历史命令窗口	Web Browser	打开网络浏览器
Current Directory	打开当前目录窗口	Array Editor	打开数组编辑器

1.3.2 命令窗口

可以直接输入命令行来实现计算或作图功能。命令窗口单独显示的效果如图 1-11 所示。

【例 1-1】简单的矩阵运算。

在命令窗口中输入以下内容：

```
>> a=[11 12 13;21 22 23;31 32 33] %创建矩阵 A
a =
    11    12    13
    21    22    23
    31    32    33
```

继续输入指令：

```
b=a*3
b =
    33    36    39
    63    66    69
    93    96    99
```

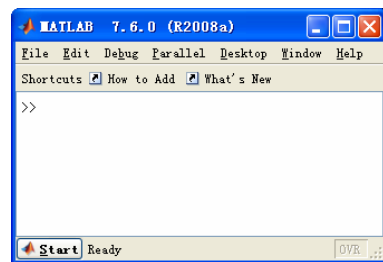


图 1-11 命令窗口单独显示的效果



(1) 每行的后面如果加上“,”，表示这行命令的计算结果不显示，否则默认每行命令所产生的变量结果都显示。

(2) MATLAB 命令窗口中可以使用连续 3 个或 3 个以上的黑点“...”来表示续行，即表示下一行是这一行的继续，在 Notebook（对 Notebook 的了解请读者参考相关资料，本书不介绍）中运行不能使用续行号。

(3) 符号“%”表示某后的内容为注意，程序将不予以计算。

通过例 1-1 可以看出命令窗口的使用方法，接下来介绍关于命令的几个知识点。

1. 输出格式的重新定义

命令窗口中数值的输出格式根据数值类型的不同显示不同的格式。当需要显示的数值为整数时，则以整数形式显示；当需要显示的数值为实数时，则以小数点后 4 位的精度近似显示，即以“短 (Short)”格式显示，如果数值的有效数字超出这一范围，则以科学计数法显示结果。

用户可以根据输出数据的显示要求来更改输出数据的显示格式。第一种方法是，单击【File】菜单下的【Preferences】命令，在弹出的【Preferences】对话框中，选择【Command Window】项，并对相应参数进行修改，如图 1-12 所示。

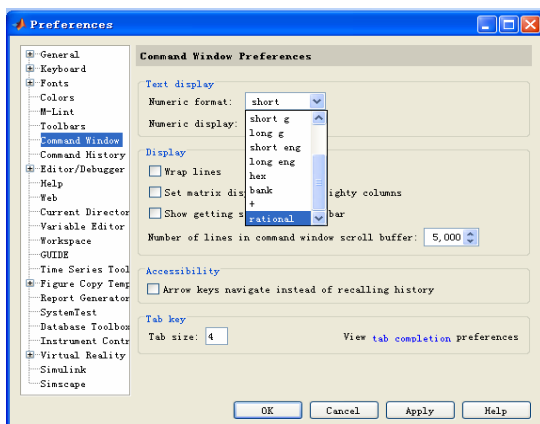


图 1-12 “Preferences”对话框

另外一个方法是在 MATLAB 工作空间中执行“format”命令，可以重新定义输出格式。MATLAB 提供的输出格式有以下几种，如表 1-3 所示。

表 1-3 数据显示格式的控制指令

指 令	含 义	指 令	含 义
format	默认值，相当于 format short	format long	用 15 位数字表示
format short	保证小数点后有 4 位有效数字	format short e	用 5 位科学计数法表示
format long e	用 15 位科学计数法表示	format rat	用近似的有理数表示
format short g	从 format short 和 format short e 中选择最佳输出方式	format compact	显示变量之间不加空行
format hex	用十六进制表示	format loose	显示变量之间加空行
format bank	用货币形式表示	format +	显示大矩阵；正数、零分别用+、-、空格表示

2. 常见的通用操作指令

MATLAB 提供了一些基本的操作指令，例如查看、保存或删除当前工作空间中的变量等，熟悉和掌握这些通用的基本操作指令，对后面的学习将会提供很大的帮助。如表 1-4 所示为常见的通用操作指令。

表 1-4 常见的通用操作指令

指 令	含 义	指 令	含 义
clc	清除命令窗口，光标回到屏幕左上角	what	列出当前目录下的 m 文件和 mat 文件
clear	从工作空间清除所有变量	clear all	从工作空间清除所有变量和函数
clf	清除图形窗口中的内容	help<命令名>	查询所列命令的帮助信息
who	列出当前工作空间中的变量	save name	保存工作空间变量到文件 name.mat
whos	列出当前工作空间中的变量及信息	save name x y	保存工作空间变量 x、y 到文件 name.mat
delete<文件名>	从磁盘中删除指定文件	load name	下载 name 文件中的所有变量到工作空间
which<文件名>	查找指定文件的路径	load name x y	下载 name 文件中的变量 x、y 到工作空间
try name.m	在工作空间查看 name.m 文件内容	save name1.m	保存工作空间一段文本到文件 name1.m

3. 命令窗口的标点符号

标点符号在 MATLAB 中具有重要的地位，各标点符号的功能如表 1-5 所示。

表 1-5 MATLAB 标点符号的功能

名 称	符 号	功 能
空格		用于输入变量之间的分隔符及数组行元素之间的分隔符
逗号	,	用于要显示计算结果的命令之间的分隔符，用于输入变量之间的分隔符，用于数组行元素之间的分隔符
点号	.	用于数值中的小数点，对于矩阵向量相乘时，表示对应位置元素相乘
冒号	:	用于生成一维数值数组，表示一维数组的全部元素或多维数组的某一维的全部元素
分号	;	用于不显示计算结果命令行的结尾，用于不显示计算结果命令行之间的分隔符，用于数组元素行之间的分隔符
单引号	''	用于括住字符串
百分号	%	用于注释的前面，在它后面的命令不需要执行
方括号	[]	用于构成向量和矩阵，用于函数输出列表
圆括号	()	用于引用数组元素，用于函数输入变量列表，用于确定算术运算的先后次序
花括号	{}	用于构成元细胞数组
续行号	...	用于把后面的行与该行连接以构成一个较长的命令
下划线	_	用于一个变量、函数或文件名中的连字符
“@”号	@	用于放在函数名前形成函数句柄，用于放在目录名前形成用户对象目录

1.3.3 当前目录浏览器窗口

当前目录浏览器窗口（Current Directory Browser）在默认情况下位于工作界面的左上方，单独显示时如图 1-13 所示。

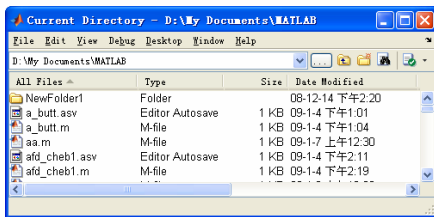


图 1-13 独立的当前目录浏览器窗口

1. 当前目录的设置

如果是通过单击桌面上的 MATLAB 图标启动，则启动后的默认当前目录是“MATLAB/work”；如果 MATLAB 是由单击“MATLAB/bin/win32”目录下的“MATLAB.exe”启动，则默认当前目录是“MATLAB/bin/win32”。

用户可以通过两种方法更改当前目录：一种是在当前目录设置区进行设置；另一种是通过命令设置。在图 1-13 中或 MATLAB 界面工具栏的右边都有当前目录设置区，在“设置栏”中直接填写待设置的目录名即可。MATLAB 提供了修改当前目录的命令 `cd`，其调用格式如下：

```
cd          %显示当前目录
cd 目录    %指定当前目录
cd ..      %指定上一级目录为当前目录
```

2. M或MAT文件描述区

MATLAB R2008 默认情况下是不显示 M 或 MAT 文件描述区的，如图 1-13 所示。如果要显示 M 或 MAT 文件描述区，单击【File】菜单下的【Preferences】菜单项，在弹出的【Preferences】对话框中单击左侧的【Current Directory】选项，在右边【Browser Display Options】中选择【Show M-file Comments and MAT-file Comments】复选框，然后单击【OK】按钮。

1.3.4 工作空间浏览器窗口

工作空间浏览器窗口（Workspace Browser）用于显示所有 MATLAB 工作空间中的变量名、数据结构、类型、大小和字节数。默认情况下位于工作界面左上侧后台，单独显示时如图 1-14 所示。

MATLAB 提供了一些管理和查看内存变量的命令，通过这些命令可以方便地完成保存、加载、删除及查看当前工作空间中的变量等操作。

1. save命令

MATLAB 中保存变量的基本命令是 `save`，使用该命令将当前工作空间中的变量以二进制的形式存储到后缀名为 MAT 的数据文件中，调用格式有如下 3 种。

① `save`：将工作空间中的所有变量数据都存放到名为 MATLAB.mat 的二进制 MAT 数据文件中。

② `save FileName`：将工作空间中的所有变量都保存到文件名为 FileName 的二进制 MAT 数据文件中。

③ `save FileName 变量1 变量2…参数`：以参数指定的保存方式将工作空间中的指定变量 1 和变量 2 等保存到文件名为 FileName 的 MAT 数据文件中。参数有“-ASCII”、“-append”等

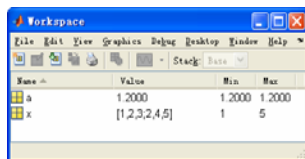


图 1-14 独立的工作空间显示窗口

方式，参数可以省略。如指定参数为“-ASCII”，则将变量 1 和变量 2 以 ASCII 文件的形式保存到名为 FileName 的 MAT 数据文件中。

2. load 命令

读取存储在文件中的变量可以使用 load 命令，该命令的使用方法与 save 类似，调用格式如下。

① load: 将 MATLAB.mat 数据文件中的变量加载到当前工作空间。

② load FileName 变量 1 变量 2...: 将指定文件中的指定变量 1、变量 2 等加载到当前工作空间，变量可以省略，省略时则加载指定文件中的所有变量。

3. 查看变量的命令

除了 save 和 load 命令，MATLAB 还提供了一些常用的查看变量的命令，如下所示。

① who: 查看 MATLAB 内存变量名。

② whos: 查看 MATLAB 内存变量名、大小、类型和字节数。

③ clear: 删除工作空间中的变量。

MATLAB 提供判断变量是否存在于工作空间的函数，调用格式如下：

i=exist('X') % 查询工作空间中是否存在变量“X”，通过返回值的不同可以得到不同的信息。返回值的具体含义如表 1-6 所示。

表 1-6 返回值的具体含义

返回值	含 义	返回值	含 义
i=0	表示不存在以下变量和文件	i=3	表示存在一个名为“X.mex”的文件
i=1	表示存在一个变量名为“X”的变量	i=4	表示存在一个名为“X.md”的文件
i=2	表示存在一个名为“X.m”的文件	i=5	表示存在一个名为“X”的内部函数

1.3.5 历史命令窗口

历史命令窗口（Command History）在默认情况下位于工作界面的左下角，单独显示的结果如图 1-15 所示。

历史命令窗口记录着用户在 MATLAB 命令窗口中输入过的全部指令行，该窗口有很多实用的功能，如表 1-7 所示。

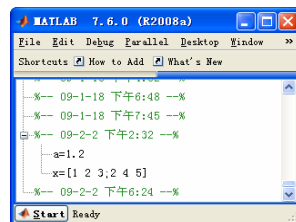


图 1-15 独立的历史命令窗口

表 1-7 历史命令窗口主要功能的操作方法

应用功能	操作方法
单行或多行命令的复制(Copy)	选中单行或多行命令，单击鼠标右键在右键菜单中选择“Copy”菜单项就可以把它复制
单行或多行命令的运行 (Evaluate Selection)	选中单行或多行命令，单击鼠标右键在右键菜单中选择“Evaluate Selection”菜单项，就可在命令窗口中运行，并得出相应结果
把多行命令写成 M 文件 (Create M-File)	选中单行或多行命令，单击鼠标右键，在菜单中选择“Create M-File”菜单项，就可以打开写有这些命令的 M 文件编辑/调试器窗口

1.3.6 数组编辑器窗口

双击工作空间浏览器窗口中的变量可以打开数组编辑器窗口 (Array Editor)，默认情况下会出现在右侧命令窗口的上方，单独显示时如图 1-16 所示。

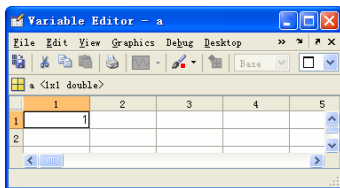


图 1-16 数组编辑器窗口

使用数组编辑器可以逐个修改数组的每个元素，并可以直接修改该数组的行数和列数。这对于建立较大数组比较有用，可以先在命令窗口中对一个新变量赋空阵，然后打开数组编辑器进行全面修改，这比直接从命令窗口中输入更为方便。

1.4 MATLAB帮助系统

MATLAB 提供了数目繁多的函数和命令，要全部把它们记下来是不现实的。可行的办法是先掌握一些基本内容，然后在实践中不断总结和积累，逐步掌握其他内容。通过软件系统本身提供的帮助功能来学习软件的使用是重要的学习方法。

MATLAB R2008 提供了丰富的帮助功能，通过这种功能可以很方便地获得有关函数和命令的使用方法。MATLAB 中通过帮助命令或帮助界面可以获得帮助。

尽管 MATLAB 的帮助文档比较实用、规范，但在使用过程中难免还会遇到一些问题，这时可以使用 MATLAB 的网上资源。在 MathWorks 公司的主页 (<http://www.mathworks.com>) 上可以找到很多有用的信息，国内一些网站也有丰富的信息资源。

1.4.1 帮助命令

要了解 MATLAB，最简捷快速的方式是在命令窗口通过帮助命令对特定的内容进行快速查询。帮助命令包括 help 命令和 lookfor 命令。

1. help命令

help 命令是查询函数语法的最基本方法，查询信息直接显示在命令窗口。在命令窗口中直接输入 help 命令将会显示当前帮助系统中所包含的所有项目，即搜索路径中所有的目录名称。

同样，可以通过 help 加函数名来显示该函数的帮助说明。例如，为了显示 magic 函数的使用方法与功能，可以使用以下命令。

```
>> help magic
```

屏幕显示帮助信息：

MAGIC Magic square.

MAGIC(N) is an N-by-N matrix constructed from the integers
1 through N^2 with equal row, column, and diagonal sums.
Produces valid magic squares for all N > 0 except N = 2.

值得注意的是，MATLAB 命令窗口里显示的帮助信息用大写来突出函数名，但在使用函数时，要用小写。

MATLAB 按照函数的不同用途分别存放在不同的子目录下，用相应的帮助命令可显示某一类函数。例如，所有的线性代数函数均收在 `matfun` 子目录下，用命令：

```
>> help matfun
```

可显示所有线性代数函数。

2. lookfor命令

`help` 命令只搜索出那些关键字完全匹配的结果，`lookfor` 命令对搜索范围内的 M 文件进行关键字搜索，条件比较宽松。例如，因为不存在 `inverse` 函数，命令：

```
help inverse
```

搜索结果为：

```
inverse not found.
```

而执行命令：

```
>> lookfor inverse
```

将得到 M 文件中包含 `inverse` 的全部函数。

`lookfor` 命令只对 M 文件的第一行进行关键字搜索。若在 `lookfor` 命令后加上 `all` 选项，则可对 M 文件进行全文搜索。

1.4.2 帮助窗口

MATLAB 帮助（`help`）窗口相当于一个帮助信息浏览器。使用帮助窗口可以搜索和查看所有 MATLAB 的帮助文档，还能运行有关演示程序。通常可以通过以下 3 种方法打开 MATLAB 帮助窗口。

- 单击 MATLAB 主窗口工具栏中的 `help` 按钮。
- 在命令窗口中运行 `helpwin`、`helpdesk` 或 `doc` 命令。
- 选择 `Help` 菜单中的 `MATLAB help` 命令。

MATLAB 帮助窗口如图 1-17 所示，该窗口包括左边的帮助向导（`Help Navigator`）窗格和右边的帮助显示窗格两部分。在左边的帮助向导窗格选择帮助项目名称或图标，将在右边的帮助显示窗格中显示对应的帮助信息。

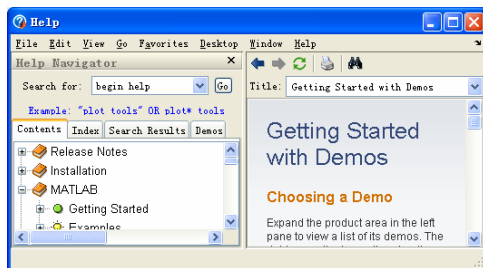


图 1-17 MATLAB 帮助窗口

在帮助向导窗格中包含有 4 个可供选择的选项卡：`Contents` 选项卡、`Index` 选项卡、`Search Results` 选项卡和 `Demos` 选项卡。`Contents` 选项卡用来查看帮助的主题；`Index` 选项卡根据指定

的关键词进行查找；Search Results 选项卡查找指定的单词；Demos 选项卡查看和运行 MATLAB 的演示程序。

MATLAB 除了有超文本格式的帮助用户文档以外，还有 PDF 格式的帮助用户文档。PDF 格式文件可用 Adobe Acrobat Reader 阅读。

1.4.3 演示系统

对于初识 MATLAB R2008 的用户，该软件自带的演示系统非常有用。要打开该系统，可以通过在帮助窗口中选择 Demos 选项卡，然后在其中选择相应的演示模块，或者在命令窗口输入 `demos`，或者选择主窗口 Help 菜单中的 Demos 命令，打开的演示系统如图 1-18 所示。

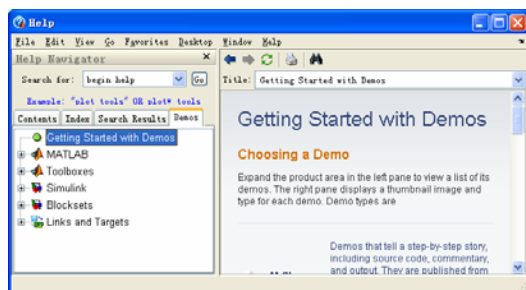


图 1-18 打开的演示系统

第2章 MATLAB数值计算及应用

数值计算是 MATLAB 中最重要、最有特色的功能之一。MATLAB 强大的数值计算功能使其成为诸多数学计算软件中的佼佼者，同时它也是 MATLAB 软件的基础。而数组和矩阵是数值计算的最基本运算单元，在 MATLAB 中，向量可以看做一维数组，而矩阵则可以看做二维数组。数组和矩阵在形式上没有区别，但二者的运算性质却有很大的不同，数组运算强调对元素的运算，而矩阵运算则采用线性代数的运算方式。

2.1 MATLAB的数值计算基础

MATLAB 的数值计算是以数组为基本单元的，而 MATLAB 数据类型的最大特点是每一种类型都以数组为基础。事实上，MATLAB 也是把每种类型的数据作为数组来处理的。

2.1.1 数据类型

数据类型是掌握任何一门编辑语言都必须首先了解的内容。MATLAB R2008 的数据类型主要有：逻辑、数值、字符串、矩阵、细胞、Java、函数句柄、稀疏以及结构等类型，其中数值型又有单精度型、双精度型以及整数型。而整数类型里面也分无符号型（uint8、uint16、uint32、uint64）和符号型（int8、int16、int32、int64）两种。在 MATLAB 中，所有的数据不管是属于什么类型，都是以数组或矩阵的形式保存的。

1. 数值型数据

数值类型包括整数型（带符号和无符号）和浮点数（单精度和双精度）两种。在默认状态下，MATLAB 将所有的数都看做是双精度的浮点数；双精度浮点数以 64 位存储，f 为 52 位，e 为 11 位，数的符号为 1 位。指数以 $e+1023$ 存储（ $1, 2^{11}-2$ ），整个浮点数的分数部分不是 f，而 $1+f$ ，IEEE 标准在 64 位中存储了 65 位的信息。所有的数值类型都支持基本的数组运算。除 int64 和 uint64 外所有的数值类型都可以应用于数学运算。

（1）整型

符号类型可以表示正数、负数和零，但是它表示的数值的范围比无符号类型要小；相反，无符号类型只能表示非负数。

【例 2-1】整型示例。

```
>> x=36.5895;
>> x=x+0
x =
    36.5895
>> x=x+0.01;
>> int16(x)      %对 x 取整
ans =
     37
>> intmin('int8') %最小的整型类数值
```



```
ans =
-128
>> intmax('int8') %最大的整型类数值
ans =
127
```

(2) 浮点型

MATLAB 中用双精度或单精度来表示浮点类型的数据，默认为双精度，但用户可以用一个简单的转换函数把任何数据用单精度来表示。

【例 2-2】浮点型示例。

```
>> clear %清除内存变量
>> x=single(30.521) %用函数 single 创建单精度数据
x =
30.5210
>> y=31.221 %创建的数据系统默认为双精度
y =
31.2210
>> whos %用 whos 函数查看数据的类型，注意 x 与 y 的类型区别
```

Name	Size	Bytes	Class	Attributes
x	1x1	4	single	
y	1x1	8	double	

(3) 复数

复数由两个独立部分组成：实部和虚部。虚部的基本单位为 $\sqrt{-1}$ ，在 MATLAB 中常用字母 i 或 j 来表示。

【例 2-3】复数示例。

```
>> clear;
>> x=rand(4)*6; %复数的创建
>> y=rand(4)*-8;
>> s=complex(x,y)
s =
4.8883 - 3.3741i 3.7942 - 5.2459i 5.7450 - 5.4299i 5.7430 - 5.2438i
5.4348 - 7.3259i 0.5852 - 0.2857i 5.7893 - 6.0619i 2.9123 - 1.3695i
0.7619 - 6.3377i 1.6710 - 6.7930i 0.9457 - 5.9451i 4.8017 - 5.6484i
5.4803 - 7.6759i 3.2813 - 7.4719i 5.8236 - 3.1378i 0.8513 - 0.2547i
```

(4) 无穷大和NaN

在 MATLAB 中，用特别的值 inf, -inf 和 NaN 分别表示无穷大、负无穷大和不确定值。

【例 2-4】无穷大和 NaN 示例。

```
>> x=log(0) %负无穷
x =
-Inf
>> y=1/0 %正无穷
y =
Inf
>> clear;
```

```
>> x=7i/0 %不确定值
```

```
x =
```

```
NaN + Inf
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
x	1x1	16	double	complex

2. 字符类型

在 MATLAB 中，字符串指的是一个统一编码的字符排列。字符串用一个向量或字符来表示，字符串存储为字符数组，每个元素占用一个 ASCII 字符，对于存储长度不一的字符串和包含多个串的数组最好使用细胞类型数组。

【例 2-5】字符类型示例。

```
>> clear
```

```
>> name='Thom R.Iee' %创建 1 行 10 列的字符数组
```

```
name =
```

```
Thom R.Iee
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
name	1x10	20	char	

```
>> name=['Thom R.Iee' ';' 'Senior Develo'] %创建二维字符数组
```

```
name =
```

```
Thom R.Iee
```

```
Senior Develo
```

3. 逻辑类型

逻辑数组类型是用数字 0 和 1 分别来表示逻辑假和逻辑真，逻辑类型的数据不一定是标量，MATLAB 也一样支持逻辑型数组，而且逻辑型的二维数组可能是稀疏的。

【例 2-6】逻辑类型示例。

```
>> x=magic(4)>=9 %创建逻辑型的数组
```

```
x =
```

1	0	0	1
0	1	1	0
1	0	0	1
0	1	1	0

4. 细胞类型

细胞类型和结构类型是 MATLAB 中比较特殊的数据类型，细胞数组提供了不同类型数据的存储机制。细胞数组的每一个元素称为一个 Cell，每一个 Cell 自己本身又是一个数组。细胞类型数组可以存储任意类型和任意维的数组。用户可以通过与矩阵和数组中同样的矩阵索引的方法来存取数据，但表示方法有所不同，如用 {1, 2} 来表示存取细胞数组的第 2 行、第 3 列的细胞。

5. 结构类型

结构是包含已命名“数据容器”或域的数组。结构类型数组中的域可以包含任意类型的数据。正如标准的数组一样，结构继承了数组的有向性特点，用户可以构建任何有效类型的大小和形状的结构数组，包括多维结构类型数组。

```
>> patient.name='Li Mi'; %创建结构类型数组 patient
>> patient.billing=130.12;
>> patient.test=[78 23 123;182 123 117;220 23.6 125.1];
>> patient
patient =
    name: 'Li Mi'
  billing: 130.1200
    test: [3x3 double]
```

6. 函数句柄类型

函数句柄用于间接调用一个函数的 MATLAB 值或数据类型。用户在调用其他函数时可以传递函数句柄，也可在数据结构中保存函数句柄以备。

```
>> fhandle=@functionname %用符号“@”构建函数句柄
fhandle =
    @functionname
>> sy=@(x)x.^2 %创建匿名函数的方法构建函数句柄
sy =
    @(x)x.^2
```

下面是一个简单函数句柄的示例。

【例 2-7】函数句柄类型示例。

%在 MATLAB 的 M-file 编辑器中输入如下代码并以 plotFhandle 名字保存：

```
function x=plotFhandle(fhandle,data)
plot(data,fhandle(data))
```

然后在命令行窗口输入命令：

```
>> plotFhandle(@cos,-pi:0.01:pi) %直接输入函数名
plot Fhandle 来调用句柄函数
```

执行程序，效果如图 2-1 所示。

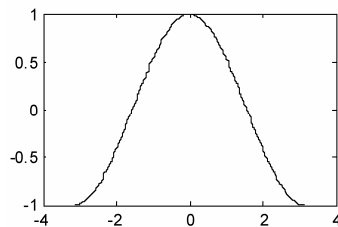


图 2-1 余弦函数图形

2.1.2 常量和变量

1. 常量

常量，在 MATLAB 中习惯称之为特殊变量，即系统自定义的变量，它们在 MATLAB 启动以后驻留在内存里面。MATLAB 常用的特殊变量如表 2-1 所示。

表 2-1 MATLAB 常用的特殊变量

特殊变量	取 值	特殊变量	取 值
ans	MATLAB 中运行结果的默认变量名	i 或 j	复数中的虚数单位, $i=j=\sqrt{-1}$
pi	圆周率 π	nargin	函数输入变量数目
eps	计算机中的最小数	nargout	函数输出变量数目
flops	浮点运算数	realmax	最大的可用正实数
inf	无穷大, 如 1/0	realmin	最小的可用正实数
NaN	不定值, 如 0/0, ∞/∞ , $0*\infty$		

在 MATLAB R2008 的命令窗口中输入一个表达式或者一组数据，系统将会自动把计算的结果赋值给“ans”变量。

【例 2-8】在命令窗口计算 $\cos(2\pi)$ 。

```
>> pi
ans =
    3.1416
>> cos(2*pi)
ans =
    1
```

2. 变量

变量是任何程序设计语言的基本要素之一，MATLAB 也不例外。与常见的程序设计语言不同的是 MATLAB 并不要求事先对所使用的变量进行声明，对变量类型进行指定，MATLAB 语言会自动根据所赋予变量的值或对变量所进行的操作来识别变量的类型并分配合适的内存空间。如果赋值变量已存在时，MATLAB 将使用新值代替旧值，同时，以新值类型代替旧值类型。如下所示。

```
>> teacher=6
```

创建一个 1×1 的名为 `teacher` 的矩阵，并将 6 作为元素的值。

2.1.3 数值计算应用的示例

下面将通过求曲线长度的例子来体会在 MATLAB 中如何使复杂的求解数学积分问题过程变得简单、明了、快捷，也直观地感受一下 MATLAB 数值计算功能的强大。

【例 2-9】求曲线长度。

曲线参数方程为

$$\begin{cases} x(t) = \sin(2t) \\ y(t) = \cos(t), t \in [0, 3\pi] \\ z(t) = t \end{cases}$$

根据曲线长度求法，我们可以列出该曲线长度的表达式

$$f(t) = \int_0^{3\pi} \sqrt{4\cos(2t)^2 + \sin(t)^2 + 1} dt$$

```
clear;
t=0:0.05:3*pi; %变量取值,步长为 0.05
plot3(sin(2*t),cos(t),t) %用 plot 函数画出曲线图形
function f=hcurve(t) %在 M-file 编辑器中编写如下代码,并保存
f=sqrt(4*cos(2*t).^2+sin(t).^2+1);
>> len=quad(@hcurve,0,3*pi) %利用积分函数 quad()求出曲线长度
len =
    17.2220
```

所求曲线的长度为 17.2220，图形如图 2-2 所示。

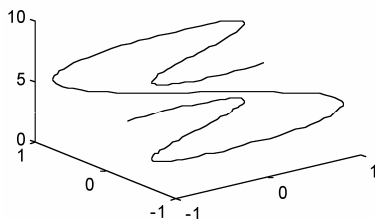


图 2-2 所求曲线的图形

2.2 MATLAB的数组、矩阵运算

2.2.1 数组与矩阵的概念

MATLAB 是“Matrix Laboratory”之意，即矩阵实验室。MATLAB 最初是为了解线性代数的问题而开发的，MATLAB 以矩阵作为基本的运算单元。矩阵是在线性代数中定义的。

线性代数中矩阵是这样定义的：有 $m \times n$ 个数 a_{ij} ($i=1,2,\dots,m; j=1,2,\dots,n$) 的数组将其排成如下格式（用方括号括起来）的表

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

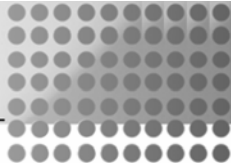
此表作为整体，将它当成一个抽象的量称为矩阵，且是 m 行 n 列的矩阵。横向每一行所有元素依次序排列则为行向量；纵向每一列所有元素依次序排列则为列向量。请特别注意，数组用方括号括起来后已作为一个抽象的特殊量——矩阵。在线性代数中，矩阵有特定的数学含义，并且有其自身严格的运算规则。矩阵概念是线性代数范畴内特有的。

在 MATLAB 中，定义了矩阵运算规则及其运算符。MATLAB 中的矩阵运算规则与线性代数中的矩阵运算规则相同。

数组 (Array) 是由一组复数排列的长方形阵列（而实数可视为复数的虚部为零的特例）。对于发展了的 MATLAB，在线性代数范畴之外，数组也是进行数值计算的基本处理单元。一行多列的数组是行向量；一行多列的数组就是列向量；数组可以是二维的“矩形”，也可以是三维的，甚至还可以是多维的。多行多列的“矩形”数组与数学中的矩阵从外观形式与数据结构上看，没有什么区别。

在 MATLAB 中，也定义了一套数组运算规则及其运算符，但数组运算是 MATLAB 软件所定义的规则，规则是为了管理数据方便、操作简单、指令形式自然、程序简单易读与运算高效。在 MATLAB 中的大量数值计算是以数组形式进行的。而在 MATLAB 中凡是涉及线性数范畴的问题，其运算则是以矩阵作为基本的运算单元。

MATLAB 既支持数组的运算也支持矩阵的运算。但在 MATLAB 中，数组与矩阵的运算却有很大的差别。在 MATLAB 中，数组的所有运算都是对运算数组中的每个元素平等地执行同样的操作。矩阵运算是从把矩阵整体当成一个特殊的量这个基点出发，依照线性代数的规则来描述的运算。



2.2.2 数组或矩阵元素的标识

为在 MATLAB 中对数组与矩阵熟练地进行运算，必须对数组与矩阵元素的定位即元素的标识非常熟悉。

1. 一维数组元素的标识、访问与赋值

一维数组（行向量）是使用方括号以及在括号内列出以空格或逗号分隔其元素的表。一维数组的元素是用数组名后圆括号内的元素在数组中位置的序号来标识的，数组元素的访问与赋值就是根据数组元素的标识进行的。

【例 2-10】一维数组元素的标识示例。

```
%在 MATLAB 命令窗口输入指令
x=[1*pi 2*pi 3*pi 4*pi 5*pi]
x =
    3.1416    6.2832    9.4248   12.5664   15.7080
%查询 x 数组的第三个元素
x(3)
ans =
    9.4248
%查询 x 数组的第 2 个到第 4 个元素
x(2:4)
ans =
    6.2832    9.4248   12.5664
%查询 x 数组的第 4 个到最后一个元素
x(4:end)
ans =
   12.5664   15.7080
%查询 x 数组的第 3、2、1 个元素
x(3:-1:1)
ans =
    9.4248    6.2832    3.1416
%查询 x 数组中小于 10 的元素
x(find(x<10))
ans =
    3.1416    6.2832    9.4248
%查询 x 数组的第 4、2、5 个元素
x([4 2 5])
ans =
   12.5664    6.2832   15.7080
%将 x 数组的第 1 个元素重新赋值为 1
x(1)=1
x =
    1.0000    6.2832    9.4248   12.5664   15.7080
```

2. 多维数组或矩阵元素的标识、访问与赋值

由于多行多列的“矩形”数组与矩阵的外观形式及数据结构相同，所以多维数组元素的标识即多维数组元素定位地址就是矩阵元素的标识或定位地址。其元素标识的通用双下标格式如下：

$a(m,n)$

其中， m 为行号； n 为列号。



有了元素的标识方法，多维数组（或矩阵）元素的访问与赋值常用的相关指令格式如表 2-2 所示。

表 2-2 子数组访问与赋值常用的相关指令格式

指令格式	功 能
<code>a(r, c)</code>	数组 <code>a</code> 中 <code>r</code> 指定行、 <code>c</code> 指定列之元素组成的子数组
<code>a(r, :)</code>	数组 <code>a</code> 中 <code>r</code> 指定行对应的所有列之元素组成的子数组
<code>a(:, c)</code>	数组 <code>a</code> 中 <code>c</code> 指定列对应的所有行之元素组成的子数组
<code>a(:)</code>	数组 <code>a</code> 的各个列按从左到右的次序首末相接的“一维长列”子数组
<code>a(i)</code>	“一维长列”子数组的第 i 个元素
<code>a(r,c)=Sa</code>	对数组 <code>a</code> 赋值, <code>Sa</code> 也必须为 <code>Sa(r, c)</code>
<code>a(:)=d(:)</code>	数组全元素赋值, 保持 <code>a</code> 的行宽、列长不变, <code>a</code> 、 <code>d</code> 两个数组元素总数应相同, 但行宽、列长可不同

【例 2-11】数组（或矩阵）元素的标识示例。

```
%查询 a 数组第 2 行、第 3 列的元素
a=[1 5 7;2 4 8;3 6 9];
a(2,3)
ans =
    8
%查询 a 数组第 3 行所有的元素
a(2,:)
ans =
    2    4    8
%查询 a 数组第 2 列转置后所有的元素
(a(:,3))'
ans =
    7    8    9
%查询 a 数组按列长转置后所有的元素
(a(:))'
ans =
    1    2    3    5    4    6    7    8    9
%查询“一维长列”a 数组第 6 个元素
a(5)
ans =
    4
%查询原 a 数组所有的元素
a
a =
    1    5    7
    2    4    8
    3    6    9
%创建 b 数组所有的元素
b=[4 4 4;5 5 5;6 6 6]
b =
    4    4    4
```

```

5     5     5
6     6     6
%以“双下标”方式对数组 a 赋值
a=b
a =
4     4     4
5     5     5
6     6     6
%创建 c 数组所有的元素
c=[7 7 7 7 7 7 7 7 7 7]
c =
7     7     7     7     7     7     7     7     7     7
%以数组全元素赋值方式对数组 a 赋值
a(:)=c(:)
a =
7     7     7
7     7     7
7     7     7

```

2.2.3 数组与矩阵的输入

一个多列的数组是行向量，矩阵横向行的所有元素依次序排列的元素也是行向量。以下介绍行向量的输入法。

1. 一维行或列向量的输入

(1) 显示元素列表输入

【例 2-12】向量元素的列表输入示例。

```

>> a=[1 2*pi sqrt(2) 4+5i]
a =
1.0000          6.2832          1.4142          4.0000 + 5.0000i
>> b=[2 5 8]'
b =
2
5
8

```

(2) 冒号生成输入

一般格式为： $x = a : \text{inc} : b$

其调用格式如例 2-13 所示。

【例 2-13】冒号生成向量示例。

```

t=0:0.1:0.6
t =
0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000

```

(3) 常量线性分隔生成法

一般格式为： $x = \text{linspace}(a, b, n)$

格式说明： a, b 分别是生成数组的第一个与最后一个元素， n 是分隔的总间隔数。这个 MATLAB 函数与指令 $x = a : (b-a)/(n-1) : b$ 等效。

【例 2-14】线性分隔生成向量示例。

```
x=linspace(0,0.6,6)
x =
    0    0.1200    0.2400    0.3600    0.4800    0.6000
```

(4) 常量对数分隔生成法

一般格式为: $x = \text{logspace}(a, b, n)$

格式功能说明: 生成数组的第一个元素为 10^a , 最后一个元素为 10^b , n 是分隔的总间隔数。

【例 2-15】对数分隔生成向量示例。

```
w=logspace(0,2,10)
w =
Columns 1 through 7
    1.0000    1.6681    2.7826    4.6416    7.7426   12.9155   21.5443
Columns 8 through 10
   35.9381   59.9484  100.0000
```

2. 二维数组（或矩阵）的输入

(1) 显示元素列表输入

在 MATLAB 中输入数组需要遵循以下基本规则:

- ① 把数组元素列入方括号[]中;
- ② 每行内的元素间用逗号或空格分开;
- ③ 行与行之间用分号或回车键(Enter)隔开。

【例 2-16】元素列表输入数组（或矩阵）示例。

```
a=[3 6 7;2 5 8;1 9 4]
a =
     3     6     7
     2     5     8
     1     9     4
```

(2) 利用 M 文件生成数组或矩阵

【例 2-17】利用 M 文件生成矩阵。

在 MATLAB 文件编辑器中输入名为 example2_1.m, 其代数如下:

```
% MATLAB File example2_1.m
e=[11 12 13;21 22 23;31 32 33]
```

在 MATLAB 中运行 example2_1.m 文件后可得到:

```
e =
    11    12    13
    21    22    23
    31    32    33
```

(3) 小矩阵连接生成大矩阵

在 MATLAB 中, 利用连接算子——方括号 ([]) 可将小矩阵连接为一个大矩阵。

【例 2-18】利用方括号 ([]) 将小矩阵连接成大矩阵示例。

```
a=[2 3;5 7];  
a1=a+7  
a2=a1+11  
a3=a2+10  
g=[a a1;a2 a3]
```

运行程序显示如下：

```
a1 =  
     9     10  
    12     14  
a2 =  
    20     21  
    23     25  
a3 =  
    30     31  
    33     35  
g =  
     2     3     9    10  
     5     7    12    14  
    20    21    30    31  
    23    25    33    35
```

由上可见，4 个 2×2 的子矩阵组成一个 4×4 的矩阵 g 。

2.2.4 数组与矩阵的算术运算

已经说明，数组无论进行什么运算，总是对被运算数组中的每一个元素进行同等的操作，矩阵运算则不同，它是把矩阵当作一个整体，依照线性代数的规则进行运算。

1. 数组或矩阵的加减运算

数组加减运算和矩阵加减运算的条件都是两个数组或矩阵的行数与列数分别相同，其运算规则也是相同的，即都是数组相应元素或矩阵相应元素的加减运算。

在 MATLAB 里，维数为 1×1 的数组叫做标量。而 MATLAB 里的数值元素是复数，所以一个标量就是一个复数。

需要指出的是，标量与数组间可以进行加减运算，其规则是标量与数组的每一个元素进行加减操作。矩阵与标量间不存在这种运算。

为了进行以下举例，先介绍著名的 Fibonacci 级数。这个级数的前两个元素为 1 与 1，第 3 个元素与以后的每个元素都是前两个元素的和。其前 9 个 Fibonacci 数为 1 1 2 3 5 8 13 21 34。

【例 2-19】前 9 个 Fibonacci 数构成的数组与标量之间实施加减运算示例。

```
clear  
K=8;  
A=[1 1 2;3 5 8;13 21 34];  
B=A+K  
C=A-K  
D=K-A
```

程序运行结果

```
B =
     9     9    10
    11    13    16
    21    29    42
C =
    -7    -7    -6
    -5    -3     0
     5    13    26
D =
     7     7     6
     5     3     0
    -5   -13   -26
```

2. 数组、矩阵的乘法运算

数组的乘法用运算符“`.*`”表示，即在乘号前加一个小点来特别指定是数组的乘法运算。数组的乘法运算必须在具有相同维数的数组之间进行，其结果是数组的对应元素间相乘的结果组成的新数组。而两矩阵相乘必须服从数学中矩阵叉乘的条件与规则。

- 数组、矩阵与标量的乘法运算：数组与一个标量之间或矩阵与一个标量之间的乘法运算都是指该数组或矩阵的每个元素与这个标量分别进行乘法运算。

【例 2-20】数组乘法运算示例。

```
clear;
k=5;
a=[1 3 5;2 6 8;11 12 13];
b=k.*a
c=k*a
d=a*k
```

运行程序结果如下：

```
b =
     5    15    25
    10    30    40
    55    60    65
c =
     5    15    25
    10    30    40
    55    60    65
d =
     5    15    25
    10    30    40
    55    60    65
```

由运行结果可知：`k.*a` 或 `k*a` 或 `a*k` 运算结果都是一样的。

- 数组、矩阵的乘法运算：数组的乘法运算必须在具有相同维数的数组之间进行，两矩阵相乘的条件是左矩阵的列数必须等于右矩阵的行数，矩阵乘法不满足交换律。

【例 2-21】数组乘法运算示例。

```
clear;  
a=[1 3 5;2 6 8;11 12 13];  
b=[7 7 7;6 6 6;2 2 2];  
c=a.*b  
d=b.*a
```

程序运行结果如下：

```
c =  
    7    21    35  
   12    36    48  
   22    24    26  
d =  
    7    21    35  
   12    36    48  
   22    24    26
```

由运行结果可知： $a.*b$ 或 $b.*a$ 运算结果都是一样的。

【例 2-22】矩阵乘法运算示例。

```
clear;  
a=[1 3 5;2 6 8;11 12 13];  
b=[6 6 6;7 7 7;1 1 1];  
c=a*b  
d=b*a
```

运行程序结果如下：

```
c =  
   32    32    32  
   62    62    62  
  163   163   163  
d =  
   84   126   156  
   98   147   182  
   14    21    26
```

由运行结果可知：矩阵乘法 $a*b$ 与 $b*a$ 两者运算结果不同，这两个矩阵乘法（ $a*b$ 与 $b*a$ ）与数组乘法运算又不同。

3. 数组、矩阵的除法运算

- 数组、矩阵与标量之间的除法运算：这里指出，标量与数组之间可以进行除法运算，其规则是标量与数组的每一个元素进行除法操作。

矩阵与标量之间无除法运算，只有矩阵右除标量（即矩阵/标量）可以运算。请看以下示例。

【例 2-23】数组、矩阵与标量之间的除法运算示例。

```
%输入程序 1  
clear;  
s=4;  
a=[1 3 5;2 4 6;3 7 9];
```

```
b=s./a
c=a.\s
d=a./s
```

运行程序结果如下：

```
b =
    4.0000    1.3333    0.8000
    2.0000    1.0000    0.6667
    1.3333    0.5714    0.4444
c =
    4.0000    1.3333    0.8000
    2.0000    1.0000    0.6667
    1.3333    0.5714    0.4444
d =
    0.2500    0.7500    1.2500
    0.5000    1.0000    1.5000
    0.7500    1.7500    2.2500
```

%输入程序 2

```
s=4;
a=[1 3 5;2 4 6;3 7 9];
e=a/s
```

运行程序结果如下：

```
e =
    0.2500    0.7500    1.2500
    0.5000    1.0000    1.5000
    0.7500    1.7500    2.2500
```

- 数组、矩阵的除法运算：数组除法与矩阵除法的运算规则也是不相同的。维数相同的两数组的除法也是对应元素之间相除，数组的除法没有左除和右除之分，即运算符“\”和“./”的运算结果是一致的，不过要注意被除数在两种除法运算符中的左右位置是不同的。

矩阵除法运算有左除与右除之分，即运算符“\”和“/”指代的运算不同。其运算规则是： $a \backslash b = \text{inv}(a) * b$, $a/b = a * \text{inv}(b)$ 。

【例 2-24】数组的除法运算示例。

```
clear;
a=[1 2 3;4 5 6;7 8 9];
b=[1 1 1;3 3 3;5 5 5];
a./b
```

运行程序结果如下：

```
ans =
    1.0000    2.0000    3.0000
    1.3333    1.6667    2.0000
    1.4000    1.6000    1.8000
```

【例 2-25】矩阵的除法运算示例。

```
c=[1 5 7;4 6 8;2 3 9];
d=[2 0 1;0 3 0;1 1 2];
a=c/d
b=inv(c)*d
e=c/d
f=c*inv(d)
g=d/c
```

运行程序结果如下：

```
a =
   -1.6667    0.2222    4.3333
         0    0.6667    4.0000
   -1.6667   -0.7778    5.3333
b =
   -0.8286    1.0571   -0.3714
    0.2857   -0.0714   -0.2857
    0.2000   -0.1000    0.4000
e =
   -1.6667    0.2222    4.3333
         0    0.6667    4.0000
   -1.6667   -0.7778    5.3333
f =
   -1.6667    0.2222    4.3333
         0    0.6667    4.0000
   -1.6667   -0.7778    5.3333
g =
   -0.8571    0.5857    0.2571
    0.8571    0.2143   -0.8571
   -0.1429    0.2143    0.1429
```

运行结果表明：第一，矩阵右除、左除是不一样的；第二，矩阵除法运算规则是 $c/d=c*\text{inv}(d)$, $c\d d=\text{inv}(c)*d$ 。

4. 数组、矩阵的乘方运算

数组的乘方使用符号“ \wedge ”来表示。

(1) 数组与标量的乘方运算

① 以数组为底而以标量为指数。

【例 2-26】以数组为底以标量为指数的乘方运算示例。

```
clear;
a=[23 33 43];
b=a.^2
b=[2 3 5;5 7 9];
c=b.^3
```

运行程序结果如下：

```
b =
    529    1089    1849
```

```
c =
      8      27     125
     125     343     729
```

运行结果表明：这种运算的规则是以数组中的每个元素为底，分别与作为指数的标量进行乘方运算得到的一个新的数组。

② 以标量为底而以数组为指数。

【例 2-27】以标量为底而以数组为指数的乘方运算示例。

```
clear;
a=[6 7 9];
b=[3 3 6;2 2 5];
d=2;
e=d.^a
f=d.^b
```

运行程序结果如下：

```
e =
     64     128     512
f =
      8      8      64
      4      4      32
```

运行结果表明：这种运算的规则是以该标量为底，用数组中的每个元素分别作为指数与该标量进行乘方运算后得到的一个新数组。

(2) 数组与数组的乘方运算示例

```
clear;
a=[2 5 6];
b=[2 4 6];
c=[1 2 3;4 5 6];
d=[9 6 3;8 5 2];
e=a.^b
f=d.^c
```

运行程序结果如下：

```
e =
      4      625     46656
f =
      9      36      27
     4096     3125      64
```

运行结果表明：数组的乘方运算规则是以前一个数组为底，后一个数组为指数，其对应的元素分别进行指数运算得到的结果。显然，数组间的乘方运算只在维数相同的数组间进行。

(3) 矩阵的乘方运算

【例 2-28】矩阵的乘方运算示例。

```
%输入程序 1
a=[1 3;4 5];
```

```
b=2;
c=[0.5];
d=-0.2;
s=a^b
```

程序 1 输出如下：

```
s =
    13    18
    24    37
%输入程序 2
AB=a^c
CD=a^d
```

程序 2 输出如下：

```
AB =
    0.6614 + 0.7500i    0.9922 - 0.3750i
    1.3229 - 0.5000i    1.9843 + 0.2500i
CD =
    0.7762 - 0.4408i   -0.0493 + 0.2204i
   -0.0657 + 0.2939i    0.7105 - 0.1469i
```

运行结果表明： a 为矩阵， c 为标量，矩阵的乘方 a^c 是矩阵 a 的 c 次方。除此之外，还可以进行标量的矩阵乘方运算与标量的数组乘方运算。

```
%输入程序 3
AC=b^a
ABC=b.^a
```

程序 3 输出如下：

```
AC =
    32.3750    47.8125
    63.7500    96.1250
ABC =
     2     8
    16    32
```

5. 数组、矩阵的转置运算

在线性代数中，把矩阵 A 的行换成同序数的列而生成的矩阵，叫做 A 的转置矩阵。从矩阵 A 生成转置矩阵的过程就是矩阵的转置运算，矩阵 A 的转置矩阵记作 A^T 。在 MATLAB 中，用运算符 “ $'$ ” 定义的矩阵转置，是其元素的共轭转置；运算符 “ $.'$ ” 定义的数组的转置则是其矩阵元素的非共轭转置。可见，线性代数定义的矩阵的转置对应着 MATLAB 中的数组转置。这是线性代数与 MATLAB 关于矩阵运算少有的区别之一。

【例 2-29】矩阵与数组的转置运算示例。

```
clear;
a=[1 4 7;2 5 8];
b=a*(1.5+i)
c=b'
g=c.'
```


运行程序结果如下：

```
b =
    1.5000 + 1.0000i    6.0000 + 4.0000i   10.5000 + 7.0000i
    3.0000 + 2.0000i    7.5000 + 5.0000i   12.0000 + 8.0000i
c =
    1.5000 - 1.0000i    3.0000 - 2.0000i
    6.0000 - 4.0000i    7.5000 - 5.0000i
   10.5000 - 7.0000i   12.0000 - 8.0000i
g =
    1.5000 - 1.0000i    6.0000 - 4.0000i   10.5000 - 7.0000i
    3.0000 - 2.0000i    7.5000 - 5.0000i   12.0000 - 8.0000i
```

2.2.5 向量及其运算

从数组的结构上看，一行多列的数组是行向量，也就是一般意义的向量，或者说向量是数组的一个特例。

1. 利用方括号生成向量

正如上述，向量是数组的一个特例，所以生成数组或矩阵的方括号用来生成向量是理所当然的。

【例 2-30】利用方括号生成向量示例。

```
a=[3 6 9 1 4 7 2 5 8]
```

显示结果如下：

```
a =
     3     6     9     1     4     7     2     5     8
```

2. 利用函数生成线性等分的向量

MATLAB 提供的函数 `linspace()` 用来生成线性等分的向量。`linspace()` 函数的调用格式有如下 2 种。

- `y=linspace(x1,x2)`：生成 100 维的向量，使 $y(1)=x_1$ ， $y(100)=x_2$ 。
- `y=linspace(x1,x2,n)`：生成 n 维的向量，使 $y(1)=x_1$ ， $y(n)=x_2$ 。

【例 2-31】利用函数 `linspace()` 来生成线性等分向量示例。

```
a=linspace(0,99,10)
a =
     0    11    22    33    44    55    66    77    88    99
```

3. 利用函数生成对数等分的问题

MATLAB 也提供了函数 `logspace()` 用来生成对数等分的向量。`logspace()` 函数的调用格式也有 2 种。

- `y=logspae(x1,x2)`：生成 50 维的数组，使 $y(1)=10^{x_1}$ ， $y(50)=10^{x_2}$ 。
- `y=logspace(x1,x2)`：生成 n 维的数组，使 $y(1)=10^{x_1}$ ， $y(n)=10^{x_2}$ 。

【例 2-32】利用函数 `logspace()` 用来生成对数等分向量示例。

```
b=logspace(0,4,5)
b =
     1    10   100  1000 10000
```

有一类三维向量，是与空间解析几何相关的，如力学、物理学中研究的力、速度与加速度。这些物理量除了有量的数值大小外，还有方向的要素。数学上将这一类向量又称为向量，以下要介绍的就是这种三维向量的几种特有的运算。

4. 向量的点积

两个三维向量的点积就是两个向量的数量积或内积，它等于两向量的模与它们之间夹角余弦的乘积。根据空间解析几何的基本原理，若两向量为

$$\mathbf{A} = \{X_1, Y_1, Z_1\} \text{ 与 } \mathbf{B} = \{X_2, Y_2, Z_2\} \quad (2-1)$$

则两个向量的点积

$$\mathbf{A} \cdot \mathbf{B} = X_1 X_2 + Y_1 Y_2 + Z_1 Z_2 \quad (2-2)$$

或

$$\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}| |\mathbf{B}| \cos \varphi \quad (2-3)$$

式中

$$|\mathbf{A}| = \sqrt{X_1^2 + Y_1^2 + Z_1^2}, \quad |\mathbf{B}| = \sqrt{X_2^2 + Y_2^2 + Z_2^2}$$

$$\cos \varphi = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| |\mathbf{B}|} = \frac{X_1 X_2 + Y_1 Y_2 + Z_1 Z_2}{\sqrt{X_1^2 + Y_1^2 + Z_1^2} \sqrt{X_2^2 + Y_2^2 + Z_2^2}} \quad (2-4)$$

MATLAB 中，提供了函数命令 `dot()` 求两个向量的数量积。函数的调用格式如下：

`C=dot(A,B)`

这种格式函数命令的功能是求两个向量 \mathbf{A} 与 \mathbf{B} 的数量积 \mathbf{C} 。需要注意，两向量 \mathbf{A} 与 \mathbf{B} 必须同是三维。

还有另一种方法求两向量 \mathbf{A} 与 \mathbf{B} 的数量积 \mathbf{C} ，即

`C=sum(A.*B)`

【例 2-33】用 MATLAB 函数计算向量 $\mathbf{a} = [1 \ 1 \ -4]$ 与 $\mathbf{b} = [2 \ -2 \ 1]$ 的点积，并用空间解析几何的基本原理公式加以验算。

程序代码如下：

```
syms x1 y1 z1 x2 y2 z2;
a=sqrt(x1^2+y1^2+z1^2);
b=sqrt(x2^2+y2^2+z2^2);
c=x1*x2+y1*y2+z1*z2;
ab=subs(c,[x1 y1 z1 x2 y2 z2],[1 1 -4 2 -2 1])
a=[1 1 -4];
b=[2 -2 1];
c=dot(a,b)
d=sum(a.*b)
```

运行程序显示如下：

```
ab =    -4
c =    -4
d =    -4
```

即 $\mathbf{a} \cdot \mathbf{b} = -4$ 且计算正确。

5. 向量的叉积

两个三维向量的叉积就是两个向量的向量积、叉乘或外积，它的定义较为复杂。设两向量 \mathbf{A} 与 \mathbf{B} ，其向量积 \mathbf{C} 表示为： $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ 。向量积 \mathbf{C} 是一个新向量，它必须满足以下 3 个条件。

① 新向量 \mathbf{C} 的模

$$|\mathbf{C}| = |\mathbf{A}||\mathbf{B}|\sin\varphi \quad (2-5)$$

即向量 \mathbf{C} 的模在数值上等于两向量 \mathbf{A} 与 \mathbf{B} 为两边的平行四边形的面积。式 (2-5) 中， φ 为两向量 \mathbf{A} 与 \mathbf{B} 之间的夹角，可以通过式 (2-4) 确定。

② 新向量 \mathbf{C} 同时垂直于向量 \mathbf{A} 与 \mathbf{B} ，即向量 \mathbf{C} 垂直于向量 \mathbf{A} 与 \mathbf{B} 所确定的平面。

③ 新向量 \mathbf{C} 正方向对应着“右手法则”，即 3 个向量 \mathbf{A} 、 \mathbf{B} 、 \mathbf{C} 附着于共同的起点，右手的拇指顺着 \mathbf{A} 的方向，食指顺着 \mathbf{B} 的方向，则 \mathbf{C} 顺着中指的方向。

MATLAB 中，提供了函数命令 `cross()` 求两个向量的向量积。函数的调用格式为：

```
C=cross(A,B)
```

这种格式的函数命令的功能是求两个向量 \mathbf{A} 与 \mathbf{B} 的向量积 \mathbf{C} 。也需要注意，两向量 \mathbf{A} 与 \mathbf{B} 必须同是三维，计算出的向量 \mathbf{C} 也是三维的。

【例 2-34】求向量 $\mathbf{a} = [1 \ 2 \ 3]$ 与 $\mathbf{b} = [4 \ 5 \ 6]$ 的点积与叉积。

其程序代码如下：

```
clear;
a=[1 2 3];
b=[2 4 5];
c=dot(a,b)
d=sum(a.*b)
e=cross(a,b)
```

运行程序显示如下：

```
c =    25
d =    25
e =
    -2     1     0
```

即 $\mathbf{a} \cdot \mathbf{b} = 26$ 与 $\mathbf{a} \times \mathbf{b} = [-2 \ 4 \ -2]$ 。

6. 向量的混合积

两个三维向量的混合积由以上两个函数实现，一般先求两个同维向量的向量积，然后再计算两个同维向量的数量积，其先后顺序不可颠倒。请看以下示例。

【例 2-35】求向量 $\mathbf{a} = [1 \ 2 \ 3]$ 与 $\mathbf{b} = [4 \ 5 \ 6]$ 的混合积。

代码如下：

```
a=[1 2 3];
b=[4 5 6];
c=cross(a,b)
d=dot(a,cross(b,c))
```

运行程序显示如下：

```
c =
    -3     6    -3
d =
    54
```

2.2.6 矩阵的特殊运算

除算术运算外，矩阵还有一些自身的特殊运算，如矩阵的行列运算、矩阵的逆运算、矩阵的秩运算、矩阵的特征值运算、矩阵的多种分解运算等。

1. 矩阵的行列式运算

由 n 阶方阵 A 的元素所构成的行列式（各元素的位置不变）称为方阵 A 的行列式，记作 $|A|$ 或 $\det A$ 。应该注意，只有方阵才能计算行列式；还要注意，方阵与行列式是两个不同的概念，方阵 A 是一个线性代数定义的特殊量——按一定方式排成的并有自身运算规则的数表； n 阶行列式则是方阵 A 的所有元素按另外的运算法则所确定的一个数，关于运算法则参考以下示例。

在 MATLAB 中，计算方阵行列式的函数命令也是 `det()`。函数的输入参数就是计算的对象方阵 A ，函数的输出参数则是计算的行列式的值。

【例 2-36】已知符号矩阵 $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ 与 $A_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ ，试计算矩阵 A 与 A_1 的行列式的值。

(1) 用以下 MATLAB 语句计算符号矩阵 A 的 2 阶行列式

```
A=sym('[a11 a12;a21 a22]')
B=det(A)
```

程序运行显示如下：

```
A =
[ a11, a12]
[ a21, a22]
B =a11*a22-a12*a21
```

即

$$\det A = \det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22} - a_{12}a_{21} \quad (2-6)$$

运行结果表明 2 阶矩阵行列式的运算规则是：从左上角到右下角的左对角线上两元素的乘积减去从右上角到左下角的右对角线上两元素的乘积。这个 2 阶矩阵行列式运算规则可以作为公式来使用。

(2) 用以下 MATLAB 语句计算符号矩阵 A_1 的 3 阶行列式

```
A1=sym('[a11 a12 a13;a21 a22 a23;a31 a32 a33]')
B1=det(A1)
```

运行程序显示如下：

```
A1 =
[ a11, a12, a13]
[ a21, a22, a23]
[ a31, a32, a33]
B1 =
a11*a22*a33-a11*a23*a32-a21*a12*a33+a21*a13*a32+a31*a12*a23-a31*a13*a22
```

即

$$B_1 = \det A_1 = \det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} =$$

$$a_{11}a_{22}a_{33} + a_{21}a_{13}a_{32} + a_{31}a_{12}a_{23} - a_{21}a_{12}a_{33} - a_{31}a_{13}a_{22} \quad (2-7)$$

对应着以上矩阵 A_1 与行列式 B_1 ，得到 3 阶矩阵行列式运算的对角线规则：左对角线上三元素的乘积、平行于左对角线的不完全左对角线上两元素与隔开左对角线的右上角和左下角元素的乘积三项取正号；右对角线上三元素的乘积、平行于右对角线的不完全右对角线上两元素与隔开右对角线的左上角和右下角元素的乘积三项取负号。这个 3 阶矩阵行列式运算规则可以作为公式来使用。

【例 2-37】试计算矩阵 $a = \begin{bmatrix} 1 & 3 & -2 \\ -1 & 2 & 4 \\ 502 & 497 & -490 \end{bmatrix}$ 与 $b = \begin{bmatrix} 9 & 5 & 8 \\ 1 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}$ 行列式的值。

程序代码如下：

```
clear;
a=[1 3 -2;-1 2 4;502 497 -490];
b=[9 5 8;1 1 2;3 2 1];
a1=det(a)
b1=det(b)
```

运行程序显示如下：

```
a1 =      4588
b1 =     -10
```

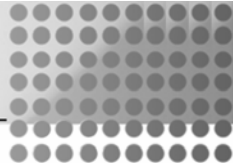
即 $a_1 = \det a = 4588$ 与 $b_1 = \det b = -10$ 。

2. 矩阵的逆运算

对于 n 阶方阵 A ，如果有一个 n 阶方阵 B 与单位阵 E ，使

$$A \cdot B = B \cdot A = E \quad (2-8)$$

则矩阵 A 是可逆的，并把矩阵 B 称为 A 的逆矩阵，记作 $B = A^{-1}$ 。求矩阵的逆矩阵，在科研与工程技术中极为广泛而重要。符号矩阵逆矩阵运算的条件是：对象 A 必为方阵且方阵 A 的行列式 $|A| \neq 0$ ，即 A 为非奇异方阵，其运算规则为



$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \mathbf{A}^* \quad (2-9)$$

式中, \mathbf{A}^* 为方阵 \mathbf{A} 的伴随矩阵。在 MATLAB 中, 可直接计算矩阵的逆矩阵, 其函数命令为 `inv()`, 其调用格式为:

`inv(A)`

函数的输入参量 \mathbf{A} 是逆矩阵运算的对象矩阵, 返回的就是对象矩阵 \mathbf{A} 的逆矩阵。

当矩阵 \mathbf{A} 为一长方形时, 可以求矩阵 \mathbf{A} 的伪逆矩阵。在 MATLAB 中, 计算长方形矩阵的伪逆矩阵的函数命令为 `pinv()`, 其调用格式为:

`pinv(A):`

函数的输入参数 \mathbf{A} 是伪逆矩阵运算的对象矩阵, 返回的就是对象矩阵 \mathbf{A} 的伪逆矩阵。

【例 2-38】试分别计算矩阵 $\mathbf{a} = \begin{bmatrix} 1 & 3 & -2 \\ -1 & 2 & 4 \\ 502 & 497 & -490 \end{bmatrix}$ 与 $\mathbf{b} = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$ 的逆矩阵与伪逆

矩阵。

其代码如下:

```
clear;
a=[1 3 -2;-1 2 4;502 497 -490];
b=[2 3 4;3 4 5;4 5 6;5 6 7];
a1=inv(a)
b1=pinv(b)
```

执行程序显示如下:

```
a1 =
   -0.6469    0.1037    0.0035
    0.3309    0.1120   -0.0004
   -0.3272    0.2199    0.0011
b1 =
   -0.9000   -0.3833    0.1333    0.6500
   -0.1000   -0.0333    0.0333    0.1000
    0.7000    0.3167   -0.0667   -0.4500
```

3. 矩阵的秩运算

矩阵秩的概念与矩阵的子式密切相关。在矩阵 \mathbf{A} 中任取 k 列, 位于这些行列交叉处的元素按原来的位置次序而得的 k 阶行列式, 叫做矩阵 \mathbf{A} 的 k 阶子式。矩阵 \mathbf{A} 的不等于 0 的子式的最高阶数称为矩阵 \mathbf{A} 的秩, 记为 $\text{rank } \mathbf{A}$ 。在 MATLAB 中求矩阵的秩的函数命令就为 `rank()`。其调用格式为:

`rank(A)`

函数的输入参量 \mathbf{A} 是求矩阵的秩的对象矩阵, 返回的就是对象矩阵 \mathbf{A} 的秩。

【例 2-39】试计算矩阵 $\mathbf{a} = \begin{bmatrix} 3 & 1 & 0 & 2 \\ 1 & -1 & 2 & -1 \\ 1 & 3 & -4 & 4 \end{bmatrix}$ 与 $\mathbf{b} = \begin{bmatrix} 3 & 2 & -1 & -3 & -1 \\ 2 & -1 & 3 & 1 & -3 \\ 7 & 0 & 5 & -1 & -8 \end{bmatrix}$ 的秩。



其代码如下：

```
clear;
a=[3 1 0 2;1 -1 2 -1;1 3 -4 4];
b=[3 2 -1 -3 -1;2 -1 3 1 -3;7 0 5 -1 -8];
c=rank(a)
d=rank(b)
```

运行程序显示如下：

```
c =      2
d =      3
```

即 $\text{rank}(a)=2$ 与 $\text{rank}(b)=3$ 。

4. 矩阵的特征值运算

矩阵的特征值与特征向量是矩阵计算的重要内容，在科学研究与工程技术中经常遇到。现将与矩阵特征值及特征向量有关的概念简介如下。

若 A 是 n 阶方阵， E 是 n 阶单位阵， A 、 E 与数 λ 构成的矩阵 $\lambda E - A$ 叫做矩阵 A 的特征矩阵。特征矩阵的行列式

$$|\lambda E - A| = \begin{vmatrix} \lambda - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \lambda - a_{nn} \end{vmatrix}$$

是 λ 的一个 n 阶多项式，叫做矩阵 A 的特征多项式。关于行列式的方程 $|\lambda E - A| = 0$ 叫做矩阵 A 的特征方程，它是一个关于 λ 的 n 次方程。特征方程 $|\lambda E - A| = 0$ 的解 λ 叫做矩阵 A 的特征根或特征值，也称为矩阵 A 的特征多项式的根；又若 λ 是 A 的特征值， θ 为零向量，则称齐次线性方程组

$$(\lambda E - A)x = \theta \quad (2-10)$$

的非零解 x 为 A 对应于特征值 λ 的特征向量。求解矩阵的特征值与特征向量又称为矩阵的特征值分解。

在 MATLAB 中，使用函数命令 `eig()` 计算矩阵的特征值与特征向量。命令 `eig()` 的调用格式有以下两种。

- `d=eig(A)`：输入参量 A 是待计算的对象矩阵，返回计算的矩阵 A 的特征值 d 是以列向量的形式给出的。如果矩阵 A 为实对称矩阵，其特征值为实数；如果矩阵 A 为非对称矩阵，其特征值为复数。
- `[V, D]=eig(A)`：输入参量同上，返回计算的矩阵 A 的特征值（对角矩阵） D 与特征向量矩阵 V ，且满足 $A \cdot V = V \cdot D$ 。

【例 2-40】试计算矩阵 $A = \begin{bmatrix} -1 & 1 & 0 \\ -4 & 3 & 0 \\ 1 & 0 & 2 \end{bmatrix}$ 的特征值、特征对角矩阵 D 与特征向量矩阵 V ，并

用式 $A \cdot V = V \cdot D$ 加以验证。

其代码如下:

```
clear;
A=[-1 1 0;-4 3 0;1 0 2];
B=eig(A)
[V,D]=eig(A)
E=A*V
F=V*D
```

执行程序显示如下:

```
B = 2
    1
    1
V =
    0    0.4082    0.4082
    0    0.8165    0.8165
  1.0000   -0.4082   -0.4082
D =
    2    0    0
    0    1    0
    0    0    1
E =
    0    0.4082    0.4082
    0    0.8165    0.8165
  2.0000   -0.4082   -0.4082
F =
    0    0.4082    0.4082
    0    0.8165    0.8165
  2.0000   -0.4082   -0.4082
```

即矩阵 A 的特征值为 $\lambda_1 = 2$ 、 $\lambda_2 = 1$ 与 $\lambda_3 = 1$, 特征值对角矩阵 $D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, 还有特征向量

矩阵 $V = \begin{bmatrix} 0 & 0.4082 & 0.4082 \\ 0 & 0.8165 & 0.8165 \\ 1.0000 & -0.4082 & -0.4082 \end{bmatrix}$, 并且 $A \cdot V = V \cdot D$ 。

5. 矩阵分解

在算术中把一个整数分解成质因数的连乘积, 在代数里将一个整式分解成几个整式因子的所谓分解因式, 这些运算是数学的基础运算之一。类似的问题在矩阵理论中称为矩阵分解, 即把一个给定的矩阵分解成几个较简单或性质比较熟悉的矩阵连乘积, 这在科学研究与工程技术的计算中都是非常重要的课题。

在 MATLAB 中, 线性方程组的求解主要基于三个基本的矩阵分解: 对称正定矩阵的分解 (Cholesky 分解)、一般方阵的消去法 (LU 分解)、矩形矩阵的正交分解 (QR 分解)。这三个分解都使用三角矩阵, 其中所有的元素位于对角线以上, 对角线以下的元素都为 0。包含三角矩阵的线性方程组使用左除或者是右除都能简单、快速地求解。

(1) 矩阵的Cholesky分解

若 A 是一个 $n \times n$ 的对称正定矩阵, 则存在对角线为正的上三角矩阵 R , 使得

$$A = R^T \cdot R \quad (2-11)$$

从 A 求 R 这就是 Cholesky 分解。在 MATLAB 中用函数命令 chol() 来实现 Cholesky 分解。函数的调用格式有以下两种。

① $R = \text{chol}(A)$: 输入参量 A 是运算对象矩阵, 输出参量 R 为上三角矩阵, 它满足 $A = R^T \cdot R$ 。如果 A 不是正定矩阵, 将给出出错信息。

② $[R, p] = \text{chol}(A)$: 输入参量同上, 输出参量 R 也同上。此时不给出出错信息, 如果 A 是正定矩阵, 则返回 $p=0$; 如果 A 不是正定矩阵, 则返回的 p 为正整数, 且上三角矩阵 R 的阶数 $n = p - 1$ 。

Cholesky 分解可用于对线性方程组 $A \cdot x = b$ 进行快速求解:

$$x = R \setminus (R^T \setminus b) \quad (2-12)$$

为了说明以下示例, 这里要介绍著名的 Pascal 矩阵。Pascal 矩阵是一对称方阵, 第一行与第一列元素都全为 1, 除 $a_{11} = 1$ 外, 其他对角线上的元素为其相邻前一行元素与其相邻前一列元素之和。

【例 2-41】Pascal 矩阵的 Cholesky 分解示例。

其代码如下:

```
clear;
a=pascal(3)
[R,p]=chol(a)
b=R.'*R
```

运行程序显示如下:

```
a =
     1     1     1
     1     2     3
     1     3     6
R =
     1     1     1
     0     1     2
     0     0     1
p =
     0
b =
     1     1     1
     1     2     3
     1     3     6
```

运算结果表明, $p = 0$, 即 Pascal 矩阵是正定矩阵; $B = A$, 即矩阵 Cholesky 分解正确。

(2) 矩阵的LU分解

将任何一个方阵 A 分解为一个下三角矩阵 L 与一个上三角矩阵 U 的乘积的运算叫做 LU 分解。即有

$$A = L \cdot U \quad (2-13)$$

在 MATLAB 中用函数命令 `lu()` 来实现 LU 分解。函数的调用格式为：

$$[L,U] = \text{lu}(A)$$

函数的输入参量 A 是对象矩阵，输出参量 L 为分解的下三角矩阵的基本变换形式（行变换）， U 为分解的上三角矩阵。 LU 分解也可用于对线性方程组 $A \cdot x = b$ 进行快速求解。

$$x = U \setminus (L \setminus b) \quad (2-14)$$

矩阵 A 行列式的值与矩阵 A 的逆还满足

$$\det(A) = \det(L) \cdot \det(U)$$

与

$$\text{inv}(A) = \text{inv}(L) \cdot \text{inv}(U)$$

【例 2-42】试对矩阵 $A = \begin{bmatrix} 38 & 2 & 14 \\ 18 & 29 & 44 \\ 41 & 47 & 5 \end{bmatrix}$ 进行 LU 分解。

其代码如下：

```
clear;
A=[38 2 14;18 29 44;41 47 5];
[L,U]=lu(A)
B=L*U
C=det(A)
D=det(L)*det(U)
E=inv(A)
F=inv(U)*inv(L)
```

运行程序显示如下：

```
L =
    0.9268    1.0000         0
    0.4390   -0.2013    1.0000
    1.0000         0         0
U =
   41.0000   47.0000    5.0000
         0  -41.5610    9.3659
         0         0   43.6901
B =
    38     2    14
    18    29    44
    41    47     5
C =
   -74448
D =
   -74448
E =
    0.0258   -0.0087    0.0043
   -0.0230    0.0052    0.0191
    0.0046    0.0229   -0.0143
```

```
F =
    0.0258   -0.0087    0.0043
   -0.0230    0.0052    0.0191
    0.0046    0.0229   -0.0143
```

运行结果表明:

- ① 输出参量 L 为下三角矩阵的基本变换形式 (行交换), U 为分解的上三角矩阵。
- ② L 与 U 满足 $\det(A)=\det(L) \cdot \det(U)$ 与 $\text{inv}(A)=\text{inv}(L) \cdot \text{inv}(U)$ 。

(3) 矩阵的QR分解

将矩形矩阵 A 分解为一个正交矩阵 Q 与一个上三角矩阵 R 的乘积的运算叫做 QR 分解, 即有

$$A = Q \cdot R \quad (2-15)$$

在 MATLAB 中, 用函数命令 `qr()` 来实现 QR 分解。函数的调用格式有以下两种。

① $[Q, R]=\text{qr}(A)$: 函数的输入参量 A 是对象矩阵, 输出参量是分解的正交矩阵 Q 与上三角矩阵 R , 满足 $A=Q \cdot R$ 。

② $[Q, R, E]=\text{qr}(A)$: 函数的输出参量 E 是一个转置矩阵, 其他同第一种格式, 此时满足

$$A \cdot E = Q \cdot R \quad (2-16)$$

【例 2-43】试对矩阵 $A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & -1 \\ 2 & -4 & 5 \end{bmatrix}$ 进行 QR 分解。

其代码如下:

```
clear;
A=[1 1 1;2 -1 -1;2 -4 5];
[Q,R,E]=qr(A)
TQ=Q.'*Q
B=Q*R
C=A*E
```

运行程序显示如下:

```
Q =
   -0.1925    0.6804   -0.7071
    0.1925   -0.6804   -0.7071
   -0.9623   -0.2722    0.0000
R =
   -5.1962    3.4641   -1.7321
         0    2.4495   -1.2247
         0         0   -2.1213
E =
     0     0     1
     0     1     0
     1     0     0
TQ =
    1.0000   -0.0000    0.0000
```

```

-0.0000    1.0000    0.0000
 0.0000    0.0000    1.0000
B =
 1.0000    1.0000    1.0000
-1.0000   -1.0000    2.0000
 5.0000   -4.0000    2.0000
C =
 1    1    1
-1   -1    2
 5   -4    2

```

运行结果表明:

① 输出参量 Q 为正交矩阵, R 为上三角矩阵;

② 分解的 Q 、 R 与 E 满足关系 $A \cdot E = Q \cdot R$.

(4) 矩阵的奇异值分解运算

矩阵的奇异值分解与前三种矩阵分解是完全不同的。对于矩阵 A 、复数域 C 与自然数 m 、 n , 若 $A \in C^{m \times n}$, 把矩阵 $A^T \cdot A$ 的 n 个特征值 $\lambda_i (i=1, 2, \dots, n)$ 的算术平方根 $\sigma_i = \sqrt{\lambda_i} (i=1, 2, \dots, n)$ 构成的矩阵叫做 A 的奇异值矩阵, 求矩阵 A 的奇异值的过程就是矩阵的奇异值分解。

在 MATLAB 中矩阵的奇异值分解用函数命令 `svd()` 来实现。函数的调用格式有以下两种。

① $S = \text{svd}(A)$: 函数的输入参量 A 是对象矩阵, 输出参量 S 是矩阵 A 的奇异值对角矩阵。

② $[U, S, V] = \text{svd}(A)$: 函数的输出参量 U 与 V 是两个正交矩阵, 且满足

$$A = U \cdot S \cdot V^T \quad (2-17)$$

U 、 S 与 V 是叫做矩阵 A 的奇异值分解三对组。

【例 2-44】试对矩阵 $A = \begin{bmatrix} 9 & 8 \\ 6 & 8 \end{bmatrix}$ 进行奇异值分解。

其代码如下:

```

clear;
A=[9 8;6 8];
ata=A'*A
[V,D]=eig(ata)
sigma=sqrt(D)
[U,S,V]=svd(A)
utu=U.*U
VTV=V.*V
usv=U*S*V'

```

运行程序显示如下:

```

ata =
 117    120
 120    128
V =
-0.7231    0.6907
 0.6907    0.7231

```

```

D =
    2.3740         0
         0  242.6260
sigma =
    1.5408         0
         0  15.5765
U =
   -0.7705   -0.6375
   -0.6375    0.7705
S =
    15.5765         0
         0    1.5408
V =
   -0.6907   -0.7231
   -0.7231    0.6907
utu =
    1.0000    0.0000
    0.0000    1.0000
VTV =
     1     0
     0     1
usv =
    9.0000    8.0000
    6.0000    8.0000

```

运行结果表明:

- ① 矩阵 $A^T \cdot A$ 的 n 个特征值 λ_i 的算术平方根 $\sigma_i = \sqrt{\lambda_i} (i=1,2,\dots,n)$ 构成的矩阵就是 A 的奇异值矩阵 S ;
- ② 参量 U 与 V 是两个正交矩阵;
- ③ U 与 V 满足 $A = U \cdot S \cdot V^T$ 。

2.2.7 数组的运算

1. 数组的关系运算

在 MATLAB 中, 关系运算与逻辑运算只适用于数组, 不适用于矩阵。数组的关系运算符已经介绍过, 现将其运算规则予以说明。

- (1) 关系运算的优先级高于算术运算, 低于逻辑运算。
- (2) 运算符 $<$ 、 $<=$ 、 $>$ 、 $>=$ 只比较两个量的实部, 而运算符 $==$ 与 $~=$ 则同时比较两个量的实部与虚部。
- (3) 若比较两个标量, 其关系成立者, 运算结果为逻辑真 (1); 否则为逻辑假 (0)。

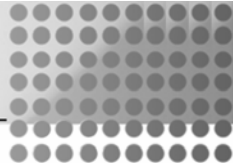
【例 2-45】数组的关系运算示例。

其代码如下:

```

a=[1+2i];
b=[4+5i];
s=a==b

```



运行程序显示如下：

```
s = 0
```

(4) 若一标量与一数组比较，则将标量与数组的每一个元素逐个比较，其运算结果为一个与数组大小（行列数）相同的数组，其元素由“1”与“0”组成，其关系成立者，运算结果为逻辑真（1）；否则为逻辑假（0）。

【例 2-46】标量与数组比较示例。

其代码如下：

```
a=9;  
b=[3 6 9;2 5 8];  
s=b<a
```

运行程序显示如下：

```
s =  
    1    1    0  
    1    1    1
```

(5) 若两数组比较，数组维数的大小（行、列数）需要相同，将两数组对应的每一个元素逐个比较，其运算结果是一个与两数组大小相同的数组，其元素由“1”与“0”组成，其关系成立者，运算结果为逻辑真（1）；否则为逻辑假（0）。

【例 2-47】数组与数组比较示例。

其代码如下：

```
e=[1 5;8 9];  
f=[3 6;5 7];  
s=e>f
```

运行程序显示如下：

```
s =  
    0    0  
    1    1
```

2. 数组的逻辑运算

数组的逻辑运算符也已经介绍过，现将其运算规则予以说明。

(1) 逻辑运算规定：非 0 元素代表逻辑真“1”；0 元素代表逻辑假“0”。

(2) 若两个标量进行逻辑与比较，两个标量全为非 0 时，运算结果为“1”，否则为“0”。

【例 2-48】两个标量逻辑与运算示例。

```
clear;  
a=1;b=1;  
c=1;d=0;  
e=0;f=1;  
g=0;h=0;  
a&b  
c&d  
e&f  
g&h
```



运行程序显示如下：

```
ans =    1
ans =    0
ans =    0
ans =    0
```

(3) 若两个标量进行逻辑或比较，两个标量全为 0 时，运算结果为“0”，否则为“1”。

【例 2-49】两个标量逻辑或运算示例。

其代码如下：

```
clear;
a=1;b=1;
c=1;d=0;
e=0;f=1;
g=0;h=0;
s0=a|b
s1=c|d
s2=e|f
s3=g|h
```

运行程序显示如下：

```
s0 =    1
s1 =    1
s2 =    1
s3 =    0
```

(4) 逻辑运算中，not 的优先级最高，and 与 or 有相同的优先级（xor 只有函数形式）；还可用括号改变运算优先权。

(5) 若一个标量与一个数组比较，则将标量与数组的每一个元素逐个比较，其运算结果为一个与数组大小（行列数）相同的数组，其元素由 1 与 0 组成。

【例 2-50】标量与数组的逻辑或运算示例。

其代码如下：

```
clear;
c=6;
d=[4 0 6;0 8 0];
s=d|c
```

运行程序显示如下：

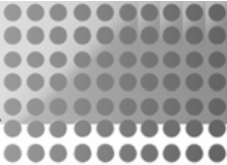
```
s =
     1     1     1
     1     1     1
```

(6) 若两个数组比较，数组维数大小（行、列数）需要相同，将两数组对应的每一个元素逐个比较，其运算结果为一个与比较数组大小相同的数组，其元素由 1 与 0 组成。

【例 2-51】数组与数组的逻辑与运算示例。

其代码如下：

```
clear;
e=[1 0;8 3];
```



```
f=[2 1;5 9];
s=e&f
```

运行程序显示如下：

```
s =
     1     0
     1     1
```

2.2.8 字符串

在 MATLAB 中，字符串作为字符数组用单引号引用到程序中。

【例 2-52】字符串作为字符数组的示例。

```
c='Hello MATLAB'
```

运行程序显示如下：

```
c =Hello MATLAB
```

请注意，变量 *c* 实际上是一个 1×9 字符数组，因为字符数组中空格也算一个字符。

在 MATLAB 里，字符是以 ASCII 数值的格式存储的，用户可以使用如下命令查看变量 *c* 在 MATLAB 内部的存储格式：

```
x=double(c)
```

回车后显示如下：

```
x =
    72    101    108    108    111    32    77    65    84    76    65    66
```

可以看到，变量 *c* 中的每个元素被转化成 ASCII 码的相应数组。

用户还可以使用函数 `char()` 将 ASCII 码的相应数字转化还原成字符。例如，输入以下指令：

```
c=char(x)
```

回车后显示如下：

```
c =Hello MATLAB
```

2.3 MATLAB 多项式及其运算

多项式运算是数学中最基本的运算之一，在许多学科里面都有着非常广泛的应用。MATLAB R2008 提供了许多多项式运算函数，如多项式的求值、求根、多项式的微积分运算、曲线拟合、插值，以及部分分式展开等，常用的多项式操作函数如表 2-3 所示。

表 2-3 常用的多项式操作函数

函 数	功能描述	函 数	功能描述
conv	多项式相乘、卷积	polyval	多项式求值
deconv	多项式相除、反卷积	polyvalm	矩阵多项式评价
poly	用多项式的根求多项式系数	residue	部分分式展开(残差运算)
polyer	多项式求导	roots	多项式求根
polyfit	多项式拟合		



2.3.1 多项式求值

MATLAB 提供的求值函数为 `polyval` 和 `polyvalm`，有些资料也称为多项式评价。

格式如下：

```
polyval(p, x)
polyvalm(p, x)
```

其中， x 可以是复数，也可以是矩阵或者数组。两个函数的区别是，前者是按照数组运算规则来计算多项式的值，而后者 x 必须为方阵，且是按照矩阵运算规则来计算多项式的值。

【例 2-53】求多项式 $f(x) = 2x^3 + 5x^2 + 7x - 2$ 在区间 $[-2, 10]$ 均匀取 100 个点的 $f(x)$ 值，并画出曲线图形。

其代码如下：

```
clear;
x=linspace(-2,10);
p=[2 5 7 -2];
v=polyval(p,x);
plot(x,v);
title('2x^3+5x^2+7x-2');
xlabel('x');
```

得到了多项式 $f(x)$ 值，如图 2-3 所示。

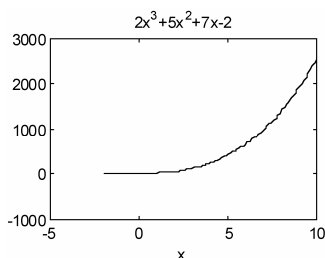


图 2-3 多项式求值

2.3.2 多项式求根

多项式求根运算在数学计算中非常常见。在 MATLAB R2008 中提供函数 `roots()` 来求根，可以通过函数 `poly()` 由多项式的根得出多项式系数。它们为一对互逆函数，在校验排序、比例化、圆整导致错误时常用到它们。

格式如下：

```
r=roots(p);
p=poly(r);
```

【例 2-54】求多项式 $f(x) = x^3 + 6x^2 + 11x + 6$ 的根，以及依据根得出多项式的系数。

其代码如下：

```
clear;
p=[1 6 11 6];
```

```

r=roots(p)
r =
    -3.0000
    -2.0000
    -1.0000
p=poly([-3 -2 -1])
p =
     1     6    11     6

```

2.3.3 部分分式展开

在信号处理和控制系统的分析应用中常常需要将分母多项式和分子多项式构成的传递函数进行部分分式展开。运用 `residue()` 函数来实现部分分式展开操作。

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \cdots + \frac{r_n}{s-p_n} + k(s)$$

格式如下：

`[r,p,k]=residue(b,a)`: `b`, `a` 分别为分子和分母多项式系数的行向量, `r` 为留数行向量
`[b,a]=residue(r,p,k)`: `p` 为极点行向量, `k` 为直接项行向量

【例 2-55】求表达式 $f(s) = \frac{5s^3 + 3s^2 - 2s + 7}{-4s^3 + 8s + 3}$ 的部分分式展开分子。

其代码如下：

```

b=[5 3 -2 7];
a=[-4 0 8 3];
[r,p,k]=residue(b,a)
[b,a]=residue(r,p,k)
r =
    -1.4167
    -0.6653
     1.3320
p =
     1.5737
    -1.1644
    -0.4093
k =
    -1.2500
b =
    -1.2500    -0.7500     0.5000    -1.7500
a =
     1.0000    -0.0000    -2.0000    -0.7500

```

所以，部分分式展开的表达式为 $f(s) = \frac{-1.25^3 - 0.75s^2 + 0.5s - 1.75}{s^3 - 2s - 0.75}$ 。

2.3.4 多项式乘除

多项式的乘法运算和除法运算在 MATLAB 中分别通过函数 `conv()` 和 `deconv()` 来实现, 同时卷积和反卷积运算也使用这两个函数。

格式如下:

```
w=conv(u,v)
[q,r]=deconv(v,u)
```

【例 2-56】 求多项式 $f(x) = 5x^3 + 6x^2 + 3x + 9$ 和 $g(x) = 7x^4 + 8x^3 + 3x^2 + 10x + 2$ 的乘积。其代码如下:

```
clear;
u=[5 6 3 9];
v=[7 8 3 10 2];
w=conv(u,v)
w =
    35    82    84   155   151    69    96    18
```

这样, 得到两个多项式相乘后的结果为:

$$T(x) = 35x^7 + 82x^6 + 84x^5 + 155x^4 + 151x^3 + 69x^2 + 96x + 18$$

【例 2-57】 求例 2-56 中两个多项式的商。

其代码如下:

```
[q,r]=deconv(u,v) %用 u 除以 v
q =
    0
r =
     5     6     3     9
[q,r]=deconv(v,u) %用 v 除以 u
q =
    1.4000   -0.0800
r =
         0         0   -0.7200   -2.3600    2.7200
conv(q,u)+r      %验算得到的结果
ans =
    7.0000    8.0000    3.0000   10.0000    2.0000
```

从例子中可以看出, 在多项式除法运算时, 不一定能够整除, 可能会有余子式。

2.3.5 多项式的微积分

在 MATLAB R2008 中有专门的函数 `polyder()` 来做多项式的微分运算, 而未提供积分运算的函数, 一般通过式子 `[p./length(p):-1:1,k]` 来进行积分运算。

格式如下:

```
m=polyder(p)
```

【例 2-58】求多项式 $f(x) = 5x^3 + 6x^2 + 3x + 9$ 微分。

其代码如下：

```
p=[5 6 3 9];
m=polyder(p)
m =
    15    12     3
```

所以，微分后得到的多项式为 $g(x) = 15x^2 + 12x + 3$ 。

【例 2-59】对例 2-58 中得到的多项式 $g(x) = 15x^2 + 12x + 3$ 求积分。

其代码如下：

```
>>p=[5 6 3 9];
m=polyder(p);
s=length(m):-1:1
s =
     3     2     1
>>p=[m./s,0]
p =
     5     6     3     0
```

所以，我们得到多项式 $g(x) = 15x^2 + 12x + 3$ 的积分是 $f(x) = 5x^3 + 6x^2 + 3x + C$ 。

2.4 插值与拟合

在已知数据中，用较简单的插值函数 $\phi(x)$ 通过所有样本点，并对临近数据进行估值计算称为插值。

插值函数 $\phi(x)$ 必须通过所有样本点。然而在有些情况下，样本点的取得本身就包含着实验中的测量误差，这一要求无疑是保留了这些测量误差的影响，满足这一要求虽然使样本点处的“误差”为零，但会使非样本点处的误差变得过大，很不合理。为此，提出了另一种函数逼近方法——数据拟合法，它不要求构造的近似函数 $\phi(x)$ 全部通过样本点，而是“很好地逼近”它们。

插值与拟合在生产和科学实验中，都有着广泛的应用。MATLAB 提供了进行插值与拟合运算的函数，可以方便地进行插值与拟合运算。

2.4.1 一维插值问题

一维插值是进行数据分析的重要手段，MATLAB 提供了 `interp1()` 函数进行一维多项式插值。`interp1()` 函数使用多项式技术，用多项式函数通过所提供的数据点，并计算目标插值点上的插值函数值，其调用格式请参看示例。

其中 `interp1()` 的插值方法有四种：

- ① `nearest`——最邻近插值。
- ② `linear`——线性插值，为默认设置。
- ③ `cubic`——三次插值。
- ④ `spline`——三次样条插值。

【例 2-60】已知的数据点来自函数，根据生成的数据进行插值处理，得出较平滑的曲线直接生成数据。

先绘制样本点图形，其代码如下：

```
x=0:0.12:1;
y=(x.^2-3*x+5).*exp(-5*x).*sin(x); %等距输入样本点
plot(x,y,'ro',x,y) %绘制样本点（已知数据点），如图 2-4（a）所示
```

可以看出，由这样的数据直接连线绘制出的曲线十分粗糙，可以再选择一组插值点，然后直接调用 `interp1()` 函数进行插值。

```
x1=0:0.02:1; %要插值点
y1=(x1.^2-3*x1+5).*exp(-5*x1).*sin(x1);
y2=interp1(x,y,x1); % 默认为线性插值
y3=interp1(x,y,x1,'spline'); %三次样条插值
y4=interp1(x,y,x1,'nearest'); %最临近插值
y5=interp1(x,y,x1,'cubic'); %三次 Hermite 插值
plot(x1,[y2' y3' y4' y5'],'.',x,y,'ro',x1,y1) %绘图比较各插值方法计算结果
legend('linear','spline','nearest','cubic','样本点','原函数');
%计算各插值方法最大计算误差
[max(abs(y1-y2)),max(abs(y1-y3)),max(abs(y1-y4)),max(abs(y1-y5))];
ans =
    0.0614    0.0086    0.1598    0.0177
```

分别选择各种插值方法，可以得出插值函数曲线与理论曲线，它们之间的比较如图 2-4（b）所示。

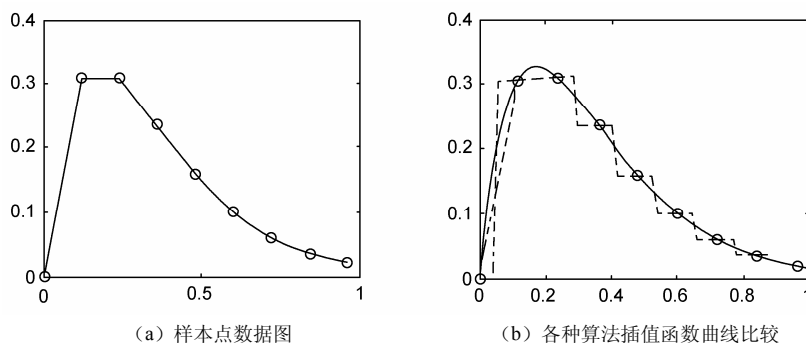


图2-4 插值函数曲线图

2.4.2 二维插值问题

二维插值是对两个自变量的插值。二维插值在图像处理和数据可视化方面有着非常重要的应用。MATLAB 提供了两个函数 `interp2()` 和 `griddata()` 来实现此功能。其中 `interp2()` 函数用于对二维网格数据进行插值；`griddata()` 函数用于二维随机数据点的插值。

1. 二维网格数据插值

MATLAB 提供了二维网格数据插值函数 `interp2()`，其用法与 `interp1()` 类似，其调用格式也请参看以下示例。

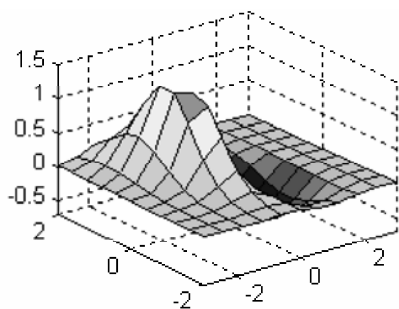
【例 2-61】由 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$ 可计算出一些较稀疏的网格数据，对整个函数曲面进行各种插值拟合，并比较插值效果。

绘制已知数据的网格图，其代码如下：

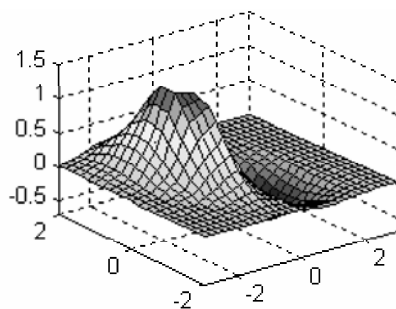
```
clear;
[x,y]=meshgrid(-3:0.6:3,-2:0.4:2);
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
surf(x,y,z); %绘制已知数据的网格图, 如图 2-5 (a) 所示
axis([-3,3,-2,2,-0.7,1.5]);
```

选择较密的插值点, 则可以用下面的 MATLAB 语句采用默认的插值算法进行插值, 得出的结果如图 2-5 (b) 所示。

```
[x1,y1]=meshgrid(-3:0.2:3,-2:0.2:2);
z1=interp2(x,y,z,x1,y1);
surf(x1,y1,z1);
axis([-3,3,-2,2,-0.7,1.5]);
```



(a) 已知数据的网格图

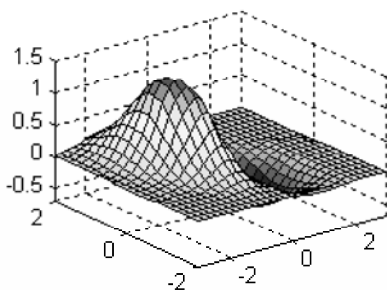


(b) 线性插值结果

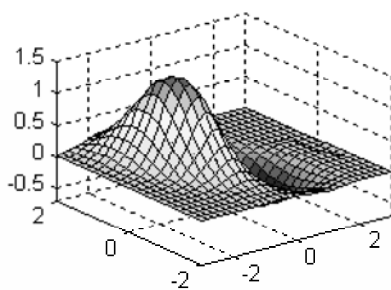
图 2-5 二维函数插值比较

可以看出, 默认的线性插值方法还原后的三维表面图在很多地方还是很粗糙。可以用下面的命令分别由立方插值选项和样条插值选项来进行插值, 得出的结果如图 2-6 所示。

```
[x1,y1]=meshgrid(-3:0.2:3,-2:0.2:2);
z2=interp2(x,y,z,x1,y1,'cubic');
surf(x1,y1,z2);
axis([-3,3,-2,2,-0.7,1.5]);
figure;
z3=interp2(x,y,z,x1,y1,'spline');
surf(x1,y1,z3);
axis([-3,3,-2,2,-0.7,1.5]);
```



(a) 立方插值算法



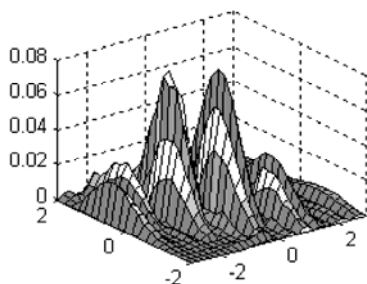
(b) 样条插值算法

图 2-6 二维函数其他插值结果比较

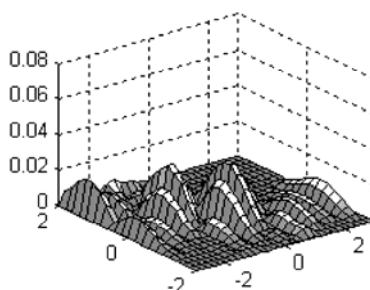
可以看出, 这样的插值结果还都是比较理想的。

通过下面的误差分析，可以对 cubic 和 spline 两种插值方法做进一步比较。因为网格已知，故可以由已知函数计算出 z 的精确值 z_0 ，可以通过下面的语句用两种算法得出矩阵 z_2 和 z_3 与真值 z_0 之间误差的绝对值，分别如图 2-7 (a) 和 2-7 (b) 所示。可以看出，选择样条方法的插值精度要远高于立方插值算法，所以在实际应用中建议选用 spline 插值选项。

```
z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1);
surf(x1,y1,abs(z0-z2));
axis([-3,3,-2,2,0,0.08]);
figure;
surf(x1,y1,abs(z0-z3));
axis([-3,3,-2,2,0,0.08]);
```



(a) 立方插值算法误差图



(b) 样条插值算法误差图

图 2-7 二维函数的误差

2. 二维随机数据点的插值

通过上面的例子可以看出，interp2 函数能够较好地进行二维插值运算。但该函数有一个重要的缺陷，就是它只能处理以网格形式给出的数据。如果已知数据不是以网格形式给出的，则该函数是无能为力的。在实际应用中，大部分问题都是以实测的多组 (x_i, y_i, z_i) 点给出的，所以不能直接使用函数 interp2 进行二维插值。

MATLAB 提供了一个更一般的 griddata() 函数，用来专门解决这样的问题。其调用格式请参考以下示例。

【例 2-62】已知二元函数 $z = f(x, y) = (x^2 - 2x)e^{-x^2 - y^2 - xy}$ ，在 $x \in [-3, 3]$ ， $y \in [-2, 2]$ 矩形区域内随机选择一组 (x_i, y_i) 坐标，就可以生成一组 z_i 的值。以这些值为已知数据，用一般分布数据插值函数 griddata() 进行插值处理，并进行误差分析。

这里选择 199 个随机数构成的点，则可以用下面的语句生成 x ， y ， z 向量，但由于这些数据不是网格数据，所以得出的数据向量不能直接用三维曲面的形式表示。但可以用下面的语句将各个样本点在 x 与 y 确定的平面上以分布的形式显示出来，如图 2-8 (a) 所示，也可以绘制出样本点的三维分布，如图 2-8 (b) 所示。可以看出，这些分布点是比较随机的。

```
clear;
x=-3+6*rand(199,1);
y=-2+4*rand(199,1);
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); %生成已知数据
plot(x,y,'*'); %样本点的二维分布
figure;
plot3(x,y,z,'*'); %样本点的三维分布
axis([-3,3,-2,2,-0.7,1.5]);
```

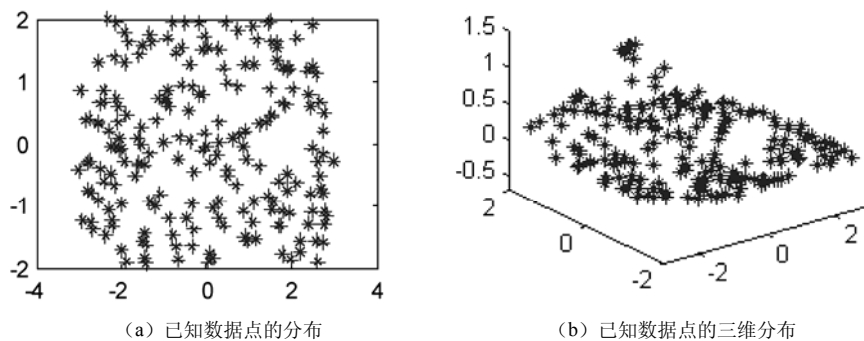


图 2-8 已知样本数据显示

用下面的语句生成网格矩阵作为插值点，用 **cubic** 和 **v4** 两种算法获得插值结果，还可以绘制出拟合后的曲面形式，分别如图 2-9 (a) 和 (b) 所示。可以看出，用 **v4** 算法得出的结果效果明显更好些。

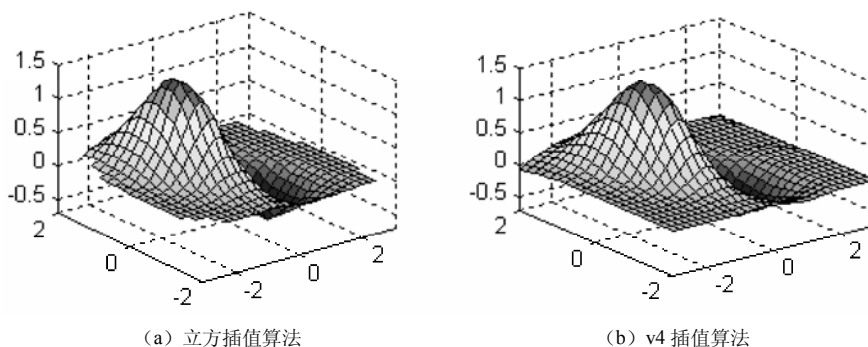


图 2-9 二维函数各种插值结果比较

```
[x1,y1]=meshgrid(-3:0.2:3,-2:0.2:2);
z1=griddata(x,y,z,x1,y1,'cubic');
surf(x1,y1,z1);
axis([-3,3,-2,2,-0.7,1.5]);
figure;
z2=griddata(x,y,z,x1,y1,'v4');
surf(x1,y1,z2);
axis([-3,3,-2,2,-0.7,1.5]);
```

还可以进一步进行误差分析。用下面的语句可以先计算出在新网格点处函数值的精确解，并用这些点和两种方法计算出来的误差，得出如图 2-10 (a) 和 (b) 所示的误差曲面。可见，用 **v4** 选项的插值结果明显优于立方插值算法，所以在实际应用中建议采用该算法。

```
z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1); %新网格点的函数值
surf(x1,y1,abs(z0-z1));
axis([-3,3,-2,2,0,0.15]);
figure;
surf(x1,y1,abs(z0-z2));
axis([-3,3,-2,2,0,0.15]);
```

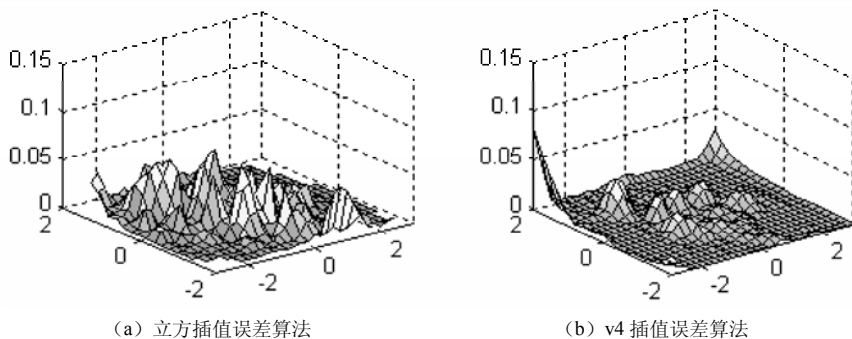



图 2-10 二维函数各种插值误差比较

2.4.3 曲线拟合

插值函数 $\phi(x)$ 必须通过所有样本点。然而,在有些情况下,样本点的取得本身就包含着实验中的测量误差,这一要求无疑是保留了这些测量误差的影响,满足这一要求虽然使样本点处“误差”为零,但会使非样本点处的误差变得过大,很不合理。为此,提出了另一种函数逼近方法——数据的拟合,它不要求构造的近似函数 $\phi(x)$ 全部通过样本点,而是“很好地逼近”它们。常用的数据拟合方法有多项式拟合和最小二乘拟合。

1. 多项式拟合

多项式拟合可以通过 MATLAB 提供的 `polyfit()` 函数实现。该函数调用格式请参看以下示例。

【例 2-63】 已知的数据点来自函数 $f(x)=1/(1+25x^2)$, $-1 \leq x \leq 1$, 根据生成的数据点进行多项式曲线拟合, 观察拟合效果。

%取不同的多项式阶次, 使用如下的语句获得多项式拟合, 并绘制出拟合曲线与原函数曲线进行对比, 如图 2-11 所示。

```
clear;
x0=-1+2*[0:10]/10;
y0=1./(1+25*x0.^2);
x=-1:0.01:1;
y1=1./(1+25*x.^2);
p2=polyfit(x0,y0,2);
y2=polyval(p2,x);
p5=polyfit(x0,y0,5);
y5=polyval(p5,x);
p8=polyfit(x0,y0,8);
y8=polyval(p8,x);
p10=polyfit(x0,y0,10);
y10=polyval(p10,x);
plot(x,y1,x,y2,'r',x,y5,'.',x,y8,'-',x,y10,'-');
legend('原函数','二次拟合','五次拟合','八次拟合','十次拟合');
```

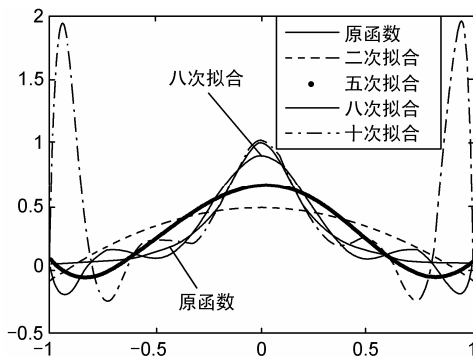


图 2-11 各阶多项式拟合效果

由该例可以看出, 多项式拟合并不是阶数越高越好, 多项式拟合的效果也并不一定是很精确的, 有时结果是相当差的, 甚至说是完全错误的。

2. 最小二乘拟合

假设有一组数据 x_i, y_i , $i=1,2,\dots,N$, 且已知这组数据满足某一函数原型 $\hat{y}(x)=f(a,x)$, 其中 a 是待定系数, 则最小二乘曲线拟合的目标就是求出这一组待定系数的值, 使得目标函数

$J = \min_a \sum_{i=1}^N [y_i - \hat{y}(x_i)]^2 = \min_a \sum_{i=1}^N [y_i - y(a, x_i)]^2$ 为最小。在 MATLAB 的最优化工具箱中提供了 lsqcurvefit() 函数, 可以解决最小二乘曲线拟合的问题。该函数调用格式请参看以下示例。

【例 2-64】已知数据 (x_i, y_i) , 满足 $y_i = 0.12e^{-0.213x_i} + 0.54e^{-0.17x_i} \sin(1.23x_i)$, 其中, $x_i = 10(i-1)/100, i=1, 2, \dots, 101$ 。并已知该数据满足函数原型 $y(x) = a_1 e^{-a_2 x} + a_3 e^{-a_4 x} \sin(a_5 x)$, 其中, a_i 为待定系数。采用最小二乘曲线拟合的目的就是获得这些待定系数, 使得目标函数的值最小。

已知函数原型, 其代码如下:

```
clear;
f=inline('a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x).*sin(a(5)*x)','a','x');
```

建立起函数的原型, 则可以由下面的语句得出待定系数向量:

```
x=0:0.1:10;
y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
[xx,res]=lsqcurvefit(f,[1 1 1 1],x,y);
xx',res
Optimization terminated: first-order optimality less than OPTIONS.TolFun, and no negative/zero
curvature detected in trust region model.
ans =
    0.1200
    0.2130
    0.5400
    0.1700
    1.2300
res =
    1.7928e-016
```

可看出, 这样得出的待定系数精度较高, 接近于理论值 $a=[0.12, 0.213, 0.54, 0.17, 1.23]^T$, 如果想进一步提高精度, 则需要修改最优化的选项, 这时函数的调用格式也将发生变化。

```
ff=optimset;
ff.TolFun=1e-20;
ff.TolX=1e-15; %修改精度限制
[xx,res]=lsqcurvefit(f,[1 1 1 1],x,y,[],[],ff);
xx',res
Optimization terminated: first-order optimality less than OPTIONS.TolFun, and no negative/zero
curvature detected in trust region model.
ans =
    0.1200
    0.2130
    0.5400
    0.1700
    1.2300
res =
    0
>> x1=0:0.1:10;
y1=f(xx,x1);
plot(x1,y1,x,y,'ro');
legend('拟合曲线','样本点');
```

其中, 两个空矩阵表示 a 向量的上下限, 由于对这些参数的范围无限制, 故采用了默认的

表示形式。可以看出，修改误差后，得出的拟合待定系数更加精确。绘制出的拟合曲线与样本点如图 2-12 所示。

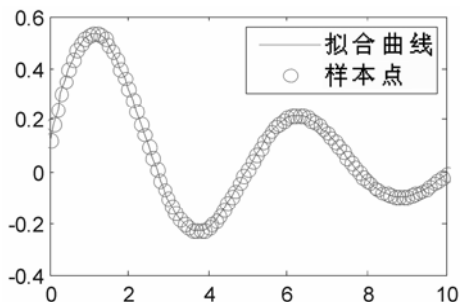


图 2-12 拟合效果比较

2.5 线性方程组求解

在 MATLAB 中，关于线性方程组的解法一般可以分为两类：一类是直接解法，就是在没有舍入误差的情况下，通过有限步的矩阵初等运算来求得方程组的解；另一类是迭代解法，就是先给定一个解的初始值，然后按照一定的迭代算法进行逐步逼近，求出更精确的近似解。

2.5.1 方程组解法

1. 直接解法

(1) 利用左除运算符的直接解法

线性方程组的直接解法大多基于高斯消元法、主元素消元法、平方根法和追赶法等。在 MATLAB 中，这些算法已经被编成了现在的库函数或运算符，因此，只需要调用相应的函数或运算符即可完成线性方程组的求解。其中，最简单的方法就是使用左除运算符“\”。程序会自动根据输入的系数矩阵判断选用哪种方法进行求解。

直接参看示例，应用直接解法求线性方程的解。

【例 2-65】用直接解法求解下列线性方程组。

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 13 \\ x_1 - 5x_2 + 7x_4 = -9 \\ 2x_2 + x_3 - x_4 = 6 \\ x_1 + 6x_2 - x_3 - 4x_4 = 0 \end{cases}$$

其代码如下：

```
clear;
a=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];
b=[13,-9,6,0]';
x=a\b
x =
    -66.5556
     25.6667
    -18.7778
     26.5556
```

(2) 利用矩阵的分解求解线性方程组

矩阵分解是指根据一定的原理用某种算法将一个矩阵分解成若干个矩阵的乘积。常见的矩阵分解有 LU 分解、QR 分解、Cholesky 分解, 以及 Schur 分解、Hessenberg 分解、奇异值分解等。这里着重介绍前 3 种常见的分解。通过这些分解方法求解线性方程组的优点是运算速度快、可以节省存储空间。

● LU 分解

矩阵的 LU 分解就是将一个矩阵表示为一个交换下三角矩阵和一个上三角矩阵的乘积形式。线性代数中已经证明, 只要矩阵 A 是非奇异的, LU 分解总是可以进行的。

MATLAB 提供的 `lu` 函数用于对矩阵进行 LU 分解, 其调用格式有两种, 请参看以下示例。

【例 2-66】利用第一种格式, 设 $A = \begin{bmatrix} 1 & -1 & 1 \\ 5 & -4 & 3 \\ 2 & 1 & 1 \end{bmatrix}$, 则对矩阵 A 进行 LU 分解的代码如下:

```
>>clear;
>>A=[1,-1,1;5,-4,3;2,1,1]
A =
     1     -1     1
     5     -4     3
     2      1     1
>>[L,U]=lu(A)
L =
    0.2000   -0.0769    1.0000
    1.0000         0         0
    0.4000    1.0000         0
U =
    5.0000   -4.0000    3.0000
         0    2.6000   -0.2000
         0         0    0.3846
```

为检验结果是否正确, 其代码如下:

```
>>LU=L*U
LU =
     1     -1     1
     5     -4     3
     2      1     1
```

说明结果是正确的。例中所获得的矩阵 L 并不是一个下三角矩阵, 但经过各行互换后, 即可获得一个下三角矩阵。

利用第二种格式对矩阵 A 进行 LU 分解:

```
>>[L,U,P]=lu(A)
L =
    1.0000         0         0
    0.4000    1.0000         0
    0.2000   -0.0769    1.0000
```

```

U =
    5.0000   -4.0000    3.0000
         0    2.6000   -0.2000
         0         0    0.3846

P =
     0     1     0
     0     0     1
     1     0     0
>>LU=L*U %这种分解其乘积不为 A
LU =
     5     -4     3
     2     1     1
     1     -1     1
>>inv(P)*L*U %考虑矩阵 P 后其乘积等于 A
ans =
     1     -1     1
     5     -4     3
     2     1     1

```

● QR 分解

对矩阵 X 进行 QR 分解，就是把 X 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积形式。QR 分解只能对方阵进行。MATLAB 的函数 `qr` 可用于对矩阵进行 QR 分解，其调用格式有两种，请参看以下示例。

利用格式一

设：

$$A = \begin{bmatrix} 1 & -1 & 1 \\ 5 & -4 & 3 \\ 2 & 7 & 10 \end{bmatrix}$$

则对矩阵 A 进行 QR 分解的代码如下：

```

clear;
A=[1,-1,1;5,-4,3;2,7,10];
[Q,R]=qr(A)
Q =
   -0.1826   -0.0956   -0.9785
   -0.9129   -0.3532    0.2048
   -0.3651    0.9307   -0.0228
R =
   -5.4772    1.2780   -6.5727
         0    8.0229    8.1517
         0         0   -0.5917

```

为检验结果是否正确，输入代码如下：

```

>> QR=Q*R
QR =

```

1.0000	-1.0000	1.0000
5.0000	-4.0000	3.0000
2.0000	7.0000	10.0000

说明结果是正确的。利用第二种格式对矩阵 A 进行 QR 分解:

```
>> [Q,R,E]=qr(A)
Q =
   -0.0953   -0.2514   -0.9632
   -0.2860   -0.9199    0.2684
   -0.9535    0.3011    0.0158
R =
  -10.4881   -5.4347   -3.4325
         0    6.0385   -4.2485
         0         0    0.4105
E =
     0     0     1
     0     1     0
     1     0     0
>> Q*R/E %验证 A=Q*R*inv(E)
ans =
   1.0000   -1.0000    1.0000
   5.0000   -4.0000    3.0000
   2.0000    7.0000   10.0000
```

● Cholesky 分解

如果矩阵 X 是对称正定的, 则 Cholesky 分解将矩阵 X 分解成一个下三角矩阵和上三角矩阵的乘积。设上三角矩阵为 R , 则下三角矩阵为其转置, 即 $X = R'R$ 。MATLAB 函数 chol(X) 用于对矩阵 X 进行 Cholesky 分解, 其也有两种调用格式, 请参看以下示例。

设

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

则对矩阵 A 进行 Cholesky 分解的代码如下:

```
clear;
A=[2,1,1;1,2,-1;1,-1,3];
R=chol(A)
R =
   1.4142    0.7071    0.7071
         0    1.2247   -1.2247
         0         0    1.0000
```

可以验证 $A = R'R$:

```
>> R'*R
ans =
```

2.0000	1.0000	1.0000
1.0000	2.0000	-1.0000
1.0000	-1.0000	3.0000

利用第二种格式对矩阵 A 进行 Cholesky 分解:

```
>> [R,p]=chol(A)
R =
    1.4142    0.7071    0.7071
         0    1.2247   -1.2247
         0         0    1.0000
p =    0
```

结果中 $p=0$, 这表示矩阵 A 是一个正定矩阵。如果试图对一个非正定矩阵进行 Cholesky 分解, 则将得出错误信息, 所以, chol 函数还可以用来判定矩阵是否为正定矩阵。

2. 迭代解法

迭代解法非常适合求解大型系数矩阵的方程组。在数值分析中, 迭代解法主要包括 Jacobi 迭代法、Gauss-Seidel 迭代法、超松弛迭代法和两步迭代法。首先用一个例子说明迭代法的思想。

为了求解线性方程组

$$\begin{cases} 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 7 \\ -2x_2 + 10x_3 = 6 \end{cases}$$

将方程改写为

$$\begin{cases} x_1 = 10x_2 - 2x_3 - 7 \\ x_2 = 10x_1 - 9 \\ x_3 = \frac{1}{10}(6 + 2x_2) \end{cases}$$

这种形式的好处是将一组 x 代入右端, 可以立即得到另一组 x 。如果两组 x 相等, 那么它就是方程组的解, 不相等时可以继续迭代。例如, 选取初值 $x_1 = x_2 = x_3 = 0$, 则经过一次迭代后, 得到 $x_1 = -7$, $x_2 = -9$, $x_3 = 0.6$, 然后再继续迭代。可以构造方程的迭代公式为

$$\begin{cases} x_1^{(k+1)} = 10x_2^{(k)} - 2x_3^{(k)} - 7 \\ x_2^{(k+1)} = 10x_1^{(k)} - 9 \\ x_3^{(k+1)} = 0.6 + 0.2x_2^{(k)} \end{cases}$$

● Jacobi 迭代法

Jacobi 迭代法的 MATLAB 函数文件 jacobi.m 如下:

```
function [y,n]=jacobi(A,b,x0,eps)
if nargin==3
    eps=1.0e-6;
elseif nargin<3
    error
```

```

    return
end
D=diag(diag(A)); %求 A 的对角矩阵
L=-tril(A,-1); %求 A 的下三角矩阵
U=-triu(A,1); %求 A 的上三角矩阵
B=D\ (L+U);
f=D\b;
y=B*x0+f;
n=1; %迭代次数
while norm(y-x0)>=eps
    x0=y;
    y=B*x0+f;
    n=n+1;
end

```

【例 2-67】用 Jacobi 迭代法求解下列线性方程组。设迭代初值为 0，迭代精度为 10^{-6} 。

$$\begin{cases} 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 7 \\ -2x_2 + 10x_3 = 6 \end{cases}$$

在命令中调用函数文件 jacobi.m，代码如下：

```

clear;
A=[10,-1,0;-1,10,-2;0,-2,10];
b=[9,7,6]';
[x,n]=jacobi(A,b,[0,0,0]',1.0e-6)
x =
    0.9958
    0.9579
    0.7916
n =    11

```

● Gauss-Serdel 迭代法

Gauss-Serdel 迭代法的 MATLAB 函数文件 gausser.m 如下：

```

function [y,n]=gausser(A,b,x0,eps)
if nargin==3
    eps=1.0e-6;
elseif nargin<3
    error
    return
end
D=diag(diag(A)); %求 A 的对角矩阵
L=-tril(A,-1); %求 A 的下三角矩阵
U=-triu(A,1); %求 A 的上三角矩阵
G=(D-L)\U;
f=(D-L)\b;
y=G*x0+f;

```



```

n=1; %迭代次数
while norm(y-x0)>=eps
    x0=y;
    y=G*x0+f;
    n=n+1;
end

```

【例 2-68】用 Gauss-Serdel 迭代法求解下列线性方程组。设迭代初值为 0，迭代精度为 10^{-6} 。

$$\begin{cases} 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 7 \\ -2x_2 + 10x_3 = 6 \end{cases}$$

在命令中调用函数文件 gausser.m，代码如下：

```

clear;
A=[10,-1,0;-1,10,-2;0,-2,10];
b=[9,7,6]';
[x,n]=gausser(A,b,[0,0,0]',1.0e-6)
x =
    0.9958
    0.9579
    0.7916
n =      7

```

由此可见，一般情况下 Gauss-Serdel 迭代比 Jacobi 迭代要收敛得快一些。但这也不是绝对的，在某些情况下，Jacobi 迭代收敛而 Gauss-Serdel 迭代却可能不收敛，看下示例。

【例 2-69】分别用 Jacobi 迭代法和 Gauss-Serdel 迭代法求解下列线性方程组，看是否收敛。

$$\begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 7 \\ 6 \end{bmatrix}$$

代码如下：

```

clear;
a=[1,2,-2;1,1,1;2,2,1];
b=[9;7;6];
[x,n]=jacobi(a,b,[0,0,0]')
x =
   -27
    26
     8
n =      4
[x,n]=gausser(a,b,[0,0,0]')
x =
   NaN
   NaN

```

NaN
n = 1012

可见对此方程, 用 Jacobi 迭代收敛, 而 Gauss-Seidel 迭代不收敛。因此, 在使用迭代法时, 要考虑算法的收敛性。

2.5.2 求线性方程组的通解

线性方程组的求解分为两类: 一类是求方程组的唯一解 (即特解); 另一类是求方程组的无穷解 (即通解)。这里对线性方程组 $Ax = b$ 的求解理论做一个归纳。

(1) 当系数矩阵 A 是一个满秩方阵时, 方程 $Ax = b$ 称为恰定方程, 方程有唯一解 $x = A^{-1}b$, 这是最基本的一种情况。一般用 $x = A \setminus b$ 求解速度更快。

(2) 当方程组右端向量 $b = 0$ 时, 方程称为齐次方程组。齐次方程组总有零解, 因此称解 $x = 0$ 为平凡解。当系数矩阵 A 的秩小于 n (n 为方程组中未知变量的个数) 时, 齐次方程组有无穷多个非平凡解, 其通解中包含 $n - \text{rank}(A)$ 个线性无关的解向量, 用 MATLAB 函数 `null(A, 'r')` 可求得基础解系。

(3) 当方程组右端向量 $b \neq 0$ 时, 系数矩阵的秩 $\text{rank}(A)$ 与其增广矩阵的秩 $\text{rank}([A, b])$ 是判断其是否有解的基本条件:

① 当 $\text{rank}(A) = \text{rank}([A, b]) = n$ 时, 方程组有唯一解 $x = A \setminus b$ 或 $x = \text{pinv}(A) * b$ 。

② 当 $\text{rank}(A) = \text{rank}([A, b]) < n$ 时, 方程组有无穷多个解, 其通解等于方程组的一个特解加上对应的齐次方程组 $Ax = 0$ 的通解。可以用 $A \setminus b$ 求得方程组的一个特解, 用 `null(A, 'r')` 求得该方程组所对应的齐次方程组的基础解系, 基础解系中包含 $n - \text{rank}(A)$ 个线性无关的解向量。

③ 当 $\text{rank}(A) < \text{rank}([A, b])$ 时, 方程组无解。

有了上面的这些讨论, 下面设计一个求解线性方程组的函数文件 `line_sol.m`

```
function [x,y]=line_sol(A,b)
[m,n]=size(A);
y=[];
if norm(b)>0 %非齐次方程组
    if rank(A)==rank([A,b])
        if rank(A)==n %有唯一解
            disp('原方程组有唯一解 x');
            x=A\b;
        else %方程组有无穷多个解, 基础解系
            disp('原方程组有无穷个解, 特解为 x, 其齐次方程组的基础解系为 y');
            x=A\b;
            y=null(A,'r');
        end
    else
        disp('方程组无解'); %方程组无解
        x=[];
    end
else %齐次方程组
    disp('原方程组有零解 x');
    x=zeros(n,1); %0 解
```

```

x=zeros(n,1)
if rank(A)<n
    disp('方程组有无穷个解，基础解系为 y'); %非 0 解
    y=null(A,'r');
end
end
end

```

下面看两个应用示例，例中调用 line_sol.m 文件来解线性方程组。

【例 2-70】求解方程组

$$\begin{cases} x_1 - 2x_2 + 3x_3 - x_4 = 1 \\ 3x_1 - x_2 + 5x_3 - 3x_4 = 2 \\ 2x_1 + x_2 + 2x_3 - 2x_4 = 3 \end{cases}$$

代码如下：

```

clear;
a=[1,-2,3,-1;3,-1,5,-3;2,1,2,-2];
b=[1;2;3];
[x,y]=line_sol(a,b)

```

输出结果为：

方程组无解

```

x =    []
y =    []

```

说明该方程组无解。

【例 2-71】求方程组的通解。

$$\begin{cases} x_1 + x_2 - 3x_3 - x_4 = 1 \\ 3x_1 - x_2 - 3x_3 + 4x_4 = 4 \\ x_1 + 5x_2 - 9x_3 - 8x_4 = 0 \end{cases}$$

代码如下：

```

clear;
format rat
a=[1,1,-3,-1;3,-1,-3,4;1,5,-9,-8];
b=[1,4,0]';
[x,y]=line_sol(a,b);
x,y
format short      %恢复默认的短格式输出

```

输出结果为：

```

原方程组有无穷个解，特解为 x，其齐次方程组的基础解系为 y
Warning: Rank deficient, rank = 2,   tol =   8.8373e-015.
> In line_sol at 11
x =

```

$$y = \begin{bmatrix} 0 \\ 0 \\ -8/15 \\ 3/5 \\ 3/2 & -3/4 \\ 3/2 & 7/4 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

所以原方程组的通解为

$$X = k_1 \begin{bmatrix} 3/2 \\ 3/2 \\ 1 \\ 0 \end{bmatrix} + k_2 \begin{bmatrix} -3/4 \\ 7/4 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -8/15 \\ 3/5 \end{bmatrix}, \text{ 其中 } k_1, k_2 \text{ 为任意常数。}$$

2.6 非线性方程与最优化问题

2.6.1 非线性方程数值求解

非线性方程的求解方法很多，常用的有牛顿迭代法，但该方法需要原方程的导数，而在实际运算中这一条件有时是不能满足的，所以又出现了弦截法、二分法等其他方法。MATLAB 提供了有关函数，用于非线性方程求解。

1. 单变量非线性方程求解

在 MATLAB 中提供了一个 `fzero` 函数，可以用来求单变量非线性方程的根。该函数的调用格式请参看以下示例。

【例 2-72】 求 $f(x) = x - \frac{1}{x} + 5$ 在 $x_0 = -5$ 和 $x_0 = 1$ 作为迭代初值时的零点。

先建立函数文件 `fx.m`;

```
function f=fz(x)
f=x-1/x+5;
```

然后调用 `fzero` 函数求根:

```
>> fzero('fz',-6) %以-6 作为迭代初值
ans =
    -5.1926
>> fzero('fz',2) %以 2 作为迭代初值
ans =
    0.1926
```

2. 非线性方程组的求解

在 MATLAB 的最优化工具箱中提供了非线性方程组的求解函数 `fsolve`，其调用格式参看以下示例。

最优化工具箱提供了 20 多个优化参数选项，用户可以使用 `optimset` 函数将它们显示出来。下面仅列出一些常用的选项。

① **Display** 选项：该选项决定函数调用中间结果为显示方式，其中 'off' 为不显示，'iter' 表示每步都显示，'final' 只显示最终结果。

② **LargeScale** 选项：表示是否用大规模问题算法，取值为 'on' 或 'off'。在求解中小型问题时，通常将该选项设置为 'off'。

③ **MaxIter** 选项：表示最大允许迭代次数，默认为 400 次。选择空矩阵则表示取默认值。

④ **TolFun** 选项：表示目标函数误差容限，选择空矩阵则表示默认值 10^{-6} 。

⑤ **TolX** 选项：表示自变量误差容限，选择空矩阵则表示默认值 10^{-6} 。

可以先用 `option=optimset` 命令来调入一组默认选项值，如果想改变其中某个参数，则可以调用 `optimset` 函数完成。例如，`optimset('Display','off')` 将设定 **Display** 选项为 'off'。也可以更直观地用结构体属性的方式设置新参数。例如，不求解大规模问题时最好用下面的语句关闭大规模问题解法选项：

```
option=optimset;
option.LargeScale='off'
```

这样可以将 **LargeScale** 选项设为 'off'。

【例 2-73】求下列方程组在 (1,1,1) 附近的解，并对结果进行验证。

$$\begin{cases} \sin x + y + z^2 e^x = 0 \\ x + y + z = 0 \\ xyz = 0 \end{cases}$$

首先建立函数文件 `myfun.m`：

```
function f=myfun(X)
x=X(1);
y=X(2);
z=X(3);
f(1)=sin(x)+y+z.^2*exp(x);
f(2)=x+y+z;
f(3)=x*y*z;
```

在给定的初值 $x_0 = 1$, $y_0 = 1$, $z_0 = 1$ 下，调用 `fsolve` 函数求方程的根：

```
>> X=fsolve('myfun',[1,1,1],optimset('Display','off'))
X =
    0.0224   -0.0224   -0.0000
>> q=myfun(X)
q =
  1.0e-006 *
   -0.5931   -0.0000    0.0006
```

可见得到了较高精度的结果。

【例 2-74】求圆和直线的两个交点。

$$\text{圆: } x^2 + y^2 + z^2 = 9$$

$$\text{直线: } \begin{cases} 3x + 5y + 6z = 0 \\ x - 3y - 6z - 1 = 0 \end{cases}$$

该问题即为求解方程组

$$\begin{cases} x^2 + y^2 + z^2 - 9 = 0 \\ 3x + 5y + 6z = 0 \\ x - 3y - 6z - 1 = 0 \end{cases}$$

使用 `fsolve` 函数求解方程组时, 必须先估计出方程组的根的大致范围。所给直线的方向数是 $(-12, 24, -14)$, 故其与球心在坐标原点的球面的交点大致是 $(-1, 1, -1)$ 和 $(1, -1, 1)$, 以这两点作为迭代初值。

先建立方程组函数文件 `fxyz.m`:

```
function f=fxyz(X)
x=X(1);
y=X(2);
z=X(3);
f(1)=x^2+y^2+z^2-9;
f(2)=3*x+5*y+6*z;
f(3)=x-3*y-6*z-1;
```

再在 MATLAB 命令窗口输入, 其代码如下:

```
>> X1=fsolve('fxyz',[-1,1,-1],optimset('Display','off')) %求第一个交点
X1 =
    -0.9508    2.4016   -1.5259
>> X2=fsolve('fxyz',[1,-1,1],optimset('Display','off')) %求第二个交点
X2 =
    1.4180   -2.3361    1.2377
```

2.6.2 无约束最优化问题求解

无约束最优化问题的一般描述为

$$\min_x f(x)$$

其中, $x = [x_1, x_2, \dots, x_n]^T$, 该数学表示的含义即求取一组 x , 使得目标函数 $f(x)$ 为最小, 故这样的问题又称为最小化问题。

在实际应用中, 许多科学研究和工程计算问题都可以归结为一个最小化问题, 如能量最小、时间最短等。MATLAB 提供了 3 个求最小值的函数, 它们的调用格式如下。

① `[x,fval]=fminbnd(filename, x1, x2, option)`: 求一元函数在 $(x1, x2)$ 区间中的极小值点 x 和最小值 `fval`。

② `[x,fval]=fminsearch(filename, x0, option)`: 基于单纯形算法求多元函数的极小值点 x 和最小值 `fval`。

③ `[x,fval]=fminunc(filename, x0,option)`: 基于拟牛顿法求多元函数的极小值点 x 和最小值 $fval$ 。

确切地说, 这里讨论的也只是局域极值的问题(全域最小问题要复杂得多)。`filename` 是定义目标函数的 M 文件名。`fminbnd` 的输入变量 $x1$ 、 $x2$ 分别表示研究区间的左、右边界。`fminsearch` 和 `fminunc` 的输入变量 $x0$ 是一个向量, 表示极值点的初值。`option` 为优化参数。当目标函数的阶数大于 2 时, 使用 `fminunc` 比 `fminsearch` 更有效, 但当目标函数高度不连续时, 使用 `fminsearch` 效果较好。

MATLAB 没有专门提供求函数最大值的函数, 但只要注意到 $-f(x)$ 在区间 (a,b) 上的最小值就是 $f(x)$ 在 (a,b) 的最大值, 所以 `fminbnd(-f, x1, x2)` 返回函数 $f(x)$ 在区间 $(x1, x2)$ 上的最大值。

【例 2-75】求函数

$$f(x) = x - \frac{1}{x} + 5$$

在区间 $(-10, 1)$ 和 $(1, 10)$ 上的最小值点。

首先建立函数文件 `fx.m`:

```
function f=fx(x)
f=x-1/x+5;
```

上述函数文件也可用一个语句函数代替:

```
f=inline('x-1/x+5')
```

再在 MATLAB 命令窗口, 输入代码:

```
>> [x,fmin]=fminbnd('fx',-10,-1) %求函数在(-10,-1)内的最小值点和最小值
x =
    -9.9999
fmin =
    -4.8999
>> fminbnd('fx',1,10)
ans =
    1.0001
```

【例 2-76】设

$$f(x, y, z) = x + \frac{y^2}{4x} + \frac{z^2}{y} + \frac{2}{z}$$

求函数 f 在 $(0.5, 0.5, 0.5)$ 附近的最小值。

建立函数文件 `fxyz2.m`:

```
function f=fxyz2(u)
x=u(1);
y=u(2);
z=u(3);
f=x+y.^2./x/4+z.^2./y+2./z;
```

在 MATLAB 命令窗口，输入代码：

```
>> [u,fmin]=fminsearch('fxyz2',[0.5 0.5 0.5])    %求函数的最小值和最小值
u =
    0.5000    1.0000    1.0000
fmin =
    4.0000
```

2.6.3 有约束最优化问题求解

有约束最优化问题的一般描述为

$$\min_{x \text{ s.t. } G(x) \leq 0} f(x)$$

其中， $x=[x_1, x_2, \dots, x_n]^T$ ，该数学表示的含义即求取一组 x ，使得目标函数 $f(x)$ 为最小，且满足约束条件 $G(x) \leq 0$ 。记号 s.t. 是英文 subject to 的缩写，表示 x 要满足后面的约束条件。

约束条件可以进一步细化为如下几种。

- ① 线性不等式约束： $Ax \leq b$ 。
- ② 线性等式约束： $A_{eq}x = b_{eq}$ 。
- ③ 非线性不等式约束： $Cx \leq 0$ 。
- ④ 非线性等式约束： $C_{eq}x = 0$ 。
- ⑤ x 的下界和上界： $L_{bnd} \leq x \leq U_{bnd}$ 。

MATLAB 最优化工具箱提供了一个 `fmincon` 函数，专门用于求解各种约束下的最优化问题。其调用格式参看以下示例。

【例 2-77】求解有约束最优化问题。

$$x \text{ s.t. } \begin{cases} \min_{x_1+0.5x_2 \geq 0.4} f(x) = 0.4x_2 + x_1^2 + x_2^2 - x_1x_2 + \frac{1}{30}x_1^3 \\ 0.5x_1 + x_2 \geq 0.5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

首先编写目标函数 M 文件 `fop.m`：

```
function f=fop(x)
f=0.4*x(2)+x(1)^2+x(2)^2-x(1)*x(2)+1/30*x(1)^3;
```

再设定约束条件，并调用 `fmincon` 函数求解此约束的最优化问题：

```
x0=[0.5;0.5];
a=[-1,-0.5;-0.5,-1];
b=[-0.4;-0.5];
lb=[0;0];
option=optimset;
option.LargeScale='off';
option.Display='off';
[x,f]=fmincon('fop',x0,a,b,[],[],lb,[],[],option)
```


输出结果为：

```
x=  
    0.3394  
    0.3303  
f=  
    0.2456
```

第3章 符号运算及应用

3.1 MATLAB符号运算基础

MATLAB 不仅拥有强大的数值运算能力，还拥有强大的符号运算能力。MATLAB 提供的符号数学工具箱可以完成几乎所有的符号运算功能，并且它可以广泛应用于数学、物理、工程力学等各个学科的科研和工程中。MATLAB 还提供了与 Maple 语言良好的接口，使得 MATLAB 的符号运算更加强大。

3.1.1 符号表达式

符号表达式是代表数学、函数、算子和变量的 MATLAB 字符串或字符串数组。不要求变量有预先确定的值，符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

MATLAB 在内部把符号表达式表示成字符串，以与数字变量或运算相区别；否则，这些符号表达式几乎完全像基本的 MATLAB 命令。如表 3-1 所示是几个符号表达式及 MATLAB 等效表达式。

表 3-1 符号表达式及 MATLAB 等效表达式

符号表达式	MATLAB 表达式	符号表达式	MATLAB 表达式
$\frac{1}{2x^n}$	'1/(2*x^n)'	$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$	M=sym('[a,b;c,d]')
$\frac{1}{y} = \sqrt{2x}$	y='1/sqrt(2*x)'	$\int_b^a \frac{x^3}{\sqrt{1-x}} dx$	f=int('x^3/sqrt(1-x)', 'a', 'b')
$\cos(x^2) - \sin(2x)$	'cos(x^2)-sin(2*x)'		

1. 符号表达式创建

(1) 创建符号常量

符号常量是不含有变量的符号表达式，符号常量常常与整数很难区分，如例 3-1 所示。

【例 3-1】创建符号常量。

```
>> a=sym('cos(2*x)') %创建符号常量
a =
cos(2*x)
>> f=simplify((3*4-2)/5+1)
f =
3
>> isstr(f) %这是字符串吗（1=yes, 0=no）
ans =
0
```

在例 3-1 中，f 代表数字型符号常数 3，但不是字符串型。

(2) 创建符号变量和符号表达式

在 MATLAB 中, 符号变量和符号表达式可以通过函数 `sym` 和 `syms` 来创建。其调用格式参看以下示例。

【例 3-2】`sym` 格式调用示例。

```
>> sym1=sym('China')
sym1 =
China
>> sym2=sym('1/10')    %创建符号变量
sym2 =
1/10
>> f=sym('a*x^3+b*x^2+c*x+d') %创建符号表达式
f =
a*x^3+b*x^2+c*x+d
```

函数 `sym` 中参数用来设置符号变量或表达式的数学特性。如 `x=sym('x','real')` 表示 `x` 为实型, 同时 `x=sym('x','unreal')` 中的 `x` 为非实型, `S=sym(A,flag)` 表示当 `A` 为 `r`、`d`、`e` 或者 `f` 格式其中之一时, `sym` 函数把它从数字向量或者矩阵形式转变成符号对象形式。

【例 3-3】`syms` 示例。

```
>> whos
Name      Size      Bytes  Class  Attributes
f         1x1         158   sym
sym1      1x1         134   sym
sym2      1x1         132   sym
```

(3) 创建符号矩阵

【例 3-4】用 `sym` 函数创建符号矩阵。

```
>> M=[a,b;c,d]
??? Undefined function or variable 'a'.
>> M='[a,b;c,d]'\n
M =
[a,b;c,d]
>> M=sym('[a,b;c,d]')
M =
[ a, b]
[ c, d]
```

`M` 以 3 种方式定义: 数字型 (如果 `a`、`b`、`c`、`d` 已预先确定)、字符串型和符号矩阵型。许多符号函数能够非常巧妙地自动将字符转变为符号表达式。但在某些情况下, 尤其是建立符号数组时, 必须用函数 `sym`, 特别地将字符串变为符号表达式。隐含形式, 例如 `diff cos(x)`, 对于那些不需要参考先前结果的简单任务最有用。但是最简单形式 (无引号) 要求一个参量, 它是一个单字符的字符串、不包含插入的空格。

【例 3-5】体会符号矩阵和字符串矩阵的不同。

```
>> clear;
a=sym('[A,B;C,D]') %符号矩阵的创建方法
a =
[ A, B]
[ C, D]
>> b='[A,B;C,D]' %表达式矩阵的创建方法
```

```
b =
A,B;C,D
>>c=[A,B;C,D] %数组矩阵的创建方法，因为数值变量并未先赋值，故出现错误信息提示
??? Undefined function or variable 'A'.
```

2. 符号表达式函数操作

MATLAB 符号函数可让用户用多种方法来操作这些表达式。

【例 3-6】符号表达式的函数操作。

```
>>diff('cos(x)')
ans =
-sin(x)
>> M=sym('[A,B;C,D]')
M =
[ A, B]
[ C, D]
>>det(M)
ans =
A*D-B*C
```

MATLAB 中函数 `function argument` 形式是与 `function('argument')` 等价的。其中，`function` 是一个函数，`argument` 是一个字符串。例如，MATLAB 可以构造 `diff cos(x)` 和 `diff('cos(x)')` 二者都意味 `diff(sym'cos(x)')`。显然第一种形式更便于输入。然而，很多时候 `sym` 是必要的。

【例 3-7】函数将字符串变成符号表达式。

```
>> diff x^2+3*x+5
ans =
2*x+3
>> diff x^2 + 3*x+5
??? Error using ==> sym.diff at 36
Do not recognize argument number 2.
```

对于符号表达式，除去 `i` 和 `j` 的小写字母外，其他字母都可以作为变量。在 MATLAB 中，可以利用函数 `symvar` 询问 MATLAB 在符号表达式中认为哪一个变量是独立变量。

【例 3-8】变量的询问。

```
>> symvar('a*x+y') %询问符号变量
ans =
'a'
'x'
'y'
>> symvar('sin(omega)') %变量'omega'不是一个单字符
ans =
'omega'
>> symvar('3*i+4*j') %i 和 j 等于-1 开根号
ans =
Empty cell array: 0-by-1
```

3.1.2 符号表达式的操作及代数运算

在 MATLAB 中，符号表达式的操作和运算主要通过符号函数来实现。所有符号函数（很少的特殊情况除外）作用到符号表达式和符号数组，并返回符号表达式或数组。其结果有时可

能看起来像一个数字，但事实上它是一个内部用字符串表示的符号表达式。本节将展开介绍符号表达式一些常见的操作和代数运算。

1. 符号表达式的操作

(1) 符号表达式中自由变量的确定

如果不能确定符号表达式中的自由符号变量，一般可以通过 MATLAB 提供的函数 `findsym` 来确定。

【例 3-9】获取符号表达式中的符号变量。

```
>> f=sym('a*x^3+b*x^2+c*x+d')
f =
a*x^3+b*x^2+c*x+d
>> findsym(f) % 获得符号表达式中所有的符号变量
ans =
a, b, c, d, x
>> syms a x y z t
>> f=sym('cos(2*pi*x)+tan(y)+sin(pi*t)')
f =
cos(2*pi*x)+tan(y)+sin(pi*t)
>> findsym(f,2) % 获得符号表达式中的两个符号变量
ans =
x,y
```

(2) 提取分子和分母

如果表达式是一个有理分式（两个多项式之比），或者可以展开为有理分式（包括那些分母为 1 的分式），可利用函数 `numden` 来提取分子或分母。例如，给定如下的表达式

$$m = x^2, \quad f = \frac{ax^2}{b-x}, \quad g = \frac{3}{2}x^2 + \frac{2}{3}x - \frac{3}{5}, \quad h = \frac{x^2+3}{2x-1} + \frac{3x}{x-1}, \quad k = \begin{cases} \frac{3}{2} \times \frac{2x+1}{3} \\ \frac{4}{x^2} \times (3x+4) \end{cases}$$

在必要时，`numden` 将表达式合并、有理化并返回所得的分子和分母。

【例 3-10】分子和分母的提取。

```
>> g=sym('3/2*x^2+2/3*x-3/5')
g =
3/2*x^2+2/3*x-3/5
>> [n,d]=numden(g)
n =
45*x^2+20*x-18
d =
30
>> h=sym('(x^2+3)/(2*x-1)+3*x/(x-1)')
h =
(x^2+3)/(2*x-1)+3*x/(x-1)
>> [n,d]=numden(h)
n =
x^3+5*x^2-3
d =
(2*x-1)*(x-1)
```

在提取各部之前,这两个表达式 g 和 h 被有理化,并变换成具有分子和分母的一个简单表达式。

【例 3-11】分子和分母数组的提取。

```
>> k=sym('[3/2,(2*x+1)/3;4/x^2,3*x+4]') % 建立一个字符数组
k =
[      3/2, (2*x+1)/3]
[      4/x^2,      3*x+4]
>> [n,d]=numden(k)
n =
[      3, 2*x+1]
[      4, 3*x+4]
d =
[      2,      3]
[ x^2,      1]
```

这个表达式 k 是符号数组, `numden` 返回两个新数组 n 和 d , 其中 n 是分子数组, d 是分母数组。如果采用 `s=numden(f)` 形式, `numden` 仅把分子返回到变量 s 中。

(3) 符号表达式的化简

MATLAB 中的符号表达式有时相当复杂, 不便于进行各种操作和书写。为此, MATLAB 提供了几个命令函数用来化简复杂的符号表达式, 主要有 `pretty`、`collect`、`expand`、`horner`、`factor`、`simplify` 和 `simple` 等。

对于同一符号表达式一般有以下 3 种表达形式。

① 多项式形式: $f(x) = x^3 - 6x^2 + 11x - 6$ 。

② 因式形式: $g(x) = (x-1)(x-2)(x-3)$ 。

③ 嵌套形式: $h(x) = x(x(x-6)+11)-6$ 。

【例 3-12】给定如下 3 个符号表达式。

```
>> f=sym('x^3-6*x^2+11*x-6');
>> g=sym('(x-1)*(x-2)*(x-3)');
>> h=sym('-6+(11+(-6+x)*x)*x');
```

通过使用 `pretty` 命令, 可以得到如下形式的符号表达式 (`pretty` 主要用于将符号表达式以常用方式显示)。

```
>> pretty(f)
      3      2
      x  - 6 x  + 11 x - 6
>> pretty(g)
      (x - 1) (x - 2) (x - 3)
>> pretty(h)
      -6 + (11 + (-6 + x) x) x
```

为了使 g 和 h 的表达式以更为简单的形式表示, 可以使用如下命令。

```
>> collect(f)
ans =
x^3-6*x^2+11*x-6
>> collect(g)
ans =
x^3-6*x^2+11*x-6
```

```
>> collect(h)
ans =
x^3-6*x^2+11*x-6
```

可见，实际 f 、 g 、 h 都是同一个符号表达式，只是书写格式不同而已。

从上面的例子可以看出，`collect` 命令主要是将表达式中同类项合并，合并后的多项式以变量幂的次数按大小依次排列。

`factor` 命令主要用于符号因式分解，其命令格式在以下示例中将涉及到。

(4) 符号表达式的替换

假设有一个以 x 为变量的符号表达式，并希望将变量转换为 y 。符号表达式的化简有时候也可以通过替换来实现。MATLAB 提供了 `subs` 函数，用来在符号表达式中进行变量替换。它们的格式为 `subs(f,new,old)`，其中 f 是符号表达式， new 和 old 是字符、字符串或其他符号表达式。“新”字符串将代替表达式 f 中各个“旧”字符串；还提供函数 `subexpr` 来替换符号表达式中的子表达式。

【例 3-13】`subs` 函数的变量替换。

```
>> f='a*x^2+b*x+c' %创建函数 f(x)
f =
a*x^2+b*x+c
>> subs(f,'s','x') %以 s 表达式替换 x 表达式
ans =
a*s^2+b*s+c
>> subs(f,'alpha','a') %以'alpha'代替表达式中的'a'
ans =
alpha*x^2+b*x+c
```

【例 3-14】用 `subexpr` 函数对子表达式进行替换。

```
>> syms a b c d W
>> [V,D]=eig([a,b;c,d])
V =
[ 1, 1]
[ -(-1/2*d+1/2*a-1/2*(d^2-2*a*d+a^2+4*b*c)^(1/2))/b,
  -(-1/2*d+1/2*a+1/2*(d^2-2*a*d+a^2+4*b*c)^(1/2))/b]
D =
[1/2*d+1/2*a+1/2*(d^2-2*a*d+a^2+4*b*c)^(1/2), 0]
[ 0, 1/2*d+1/2*a-1/2*(d^2-2*a*d+a^2+4*b*c)^(1/2)]
>> [R,W]=subexpr([V;D],W)
R =
[ 1, 1]
[ -(-1/2*d+1/2*a-1/2*W)/b, -(-1/2*d+1/2*a+1/2*W)/b]
[ 1/2*d+1/2*a+1/2*W, 0]
[ 0, 1/2*d+1/2*a-1/2*W]
W =
(d^2-2*a*d+a^2+4*b*c)^(1/2)
```

(5) 符号函数的求反和复合

MATLAB 具有对符号表达式执行更高级运算的功能。对于函数 $f(x)$ ，存在另一函数 $g(x)$ 使得函数 $g(f(x)) = x$ 成立，则函数 $g(x)$ 称为函数 $f(x)$ 的反函数。在 MATLAB 中提供函数

`finverse` 来求符号函数的反函数。而用函数 `compose` 把 $f(x)$ 和 $g(x)$ 复合成 $f(g(x))$ ，以及用函数 `symsum` 求表达式的符号和。

- 符号表达式函数 $f(x)$ 的反函数 $g(x)$ ，满足 $g(f(x)) = x$ 。例如，函数 e^x 的反函数是 $\ln(x)$ ，因为 $\ln(e^x) = x$ 。函数 $\sin(x)$ 的反函数是 $\arcsin(x)$ ，函数 $\frac{1}{\tan(x)}$ 的反函数是 $\arcsin\left(\frac{1}{x}\right)$ 。函数 `finverse` 返回表达式的反函数，如果解不是唯一的，就给出警告信息。

【例 3-15】求函数的反函数。

```
>> finverse(sym('1/x'))
ans =
1/x
>> finverse(sym('x^2'))
Warning: finverse(x^2) is not unique.
> In sym.finverse at 48
ans =
x^(1/2)
>> finverse(sym('a*x+b'))
ans =
-(b-x)/a
>> finverse(sym('a*b+c*d-a*z')),sym('a')
ans =
(a*b+c*d-z)/a
ans =
a
```

- 在 MATLAB 中用函数 `compose` 把 $f(x)$ 和 $g(x)$ 复合成 $g(f(x)) = x$ 。

【例 3-16】复合下面给定的符号表达式。

$$f = \frac{1}{1+x^2}, \quad g = \sin(x), \quad h = \frac{1}{1+u^2}, \quad k = \sin(v)$$

```
>> f=sym('1/(1+x^2)');           %创建 4 个表达式
>> g=sym('sin(x)');
>> h=sym('1/(1+u^2)');
>> k=sym('sin(v)');
>> compose(f,g)                   %询问 f(g(x))表达式
ans =
1/(1+sin(x)^2)
>> compose(g,f)                   %询问 g(f(x))表达式
ans =
sin(1/(1+x^2))
```

`compose` 也可用于含有不同独立变量的函数表达式。

```
>> compose(h,k,'u','v')
ans =
1/(1+sin(v)^2)
```


- 用 `symsum` 函数求表达式的符号和有 4 种形式：`symsum(f)` 返回 $\sum_0^{x-1} f(x)$ ；`symsum(f,'s')` 返回 $\sum_0^{s-1} f(s)$ ；`symsum(f, a, b)` 返回 $\sum_a^b f(x)$ ；最普通的形式 `symsum(f, 's', a, b)` 返回 $\sum_a^b f(s)$ 。

【例 3-17】对于 $\sum_0^{x-1} x^2$ ，应返回 $\frac{x^3}{3} - \frac{x^2}{2} + \frac{x}{6}$ 。

```
>> symsum(sym('x^2'))
ans =
1/3*x^3-1/2*x^2+1/6*x
```

【例 3-18】对于 $\sum_1^n (2n-1)^2$ ，返回 $\frac{n(2n-1)(2n+1)}{3}$ 。

```
>> symsum(sym('(2*n-1)^2'),1,'n')
ans =
11/3*n+8/3-4*(n+1)^2+4/3*(n+1)^3
>> factor(ans) % 因式分解 ans
ans =
1/3*n*(2*n-1)*(2*n+1)
```

【例 3-19】对于 $\sum_1^\infty \frac{1}{(2n-1)^2}$ 应返回 $\frac{\pi^2}{8}$ 。

```
>> symsum(sym('1/(2*n-1)^2'),1,inf)
ans =
1/8*pi^2
```

(6) 符号对象的数值转换

在 MATLAB 中有 3 种不同的数据类型：数值、符号和字符。每种数据类型都有自己独特的操作函数和指令，为了能够交换不同的数据类型，MATLAB 提供一些函数来实现这一操作。比如符号表达式变换成数值或反之。有极少数的符号函数可返回数值。然而，某些符号函数能自动地将一个数字变换成它的符号表达式，如果该数字是函数许多参量中的一个。

函数 `sym` 可获取一个数字参量并将其转换为符号表达式。函数 `eval` 可将字符串传给 MATLAB 以便计算，所以 `eval` 是另一个可用于把符号常数变换为数字或计算表达式的函数。

【例 3-20】符号常数变成数值。

```
>> ph=sym('(1+sqrt(5))/2') %全局量
ph =
(1+sqrt(5))/2
>> eval(ph) %转换成数值型
ans =
1.6180
```

符号函数 `sym2poly` 将符号多项式变换成它的 MATLAB 等价系数向量。函数 `poly2sym` 功能正好相反，并让用户指定用于所得结果表达式中的变量。

【例 3-21】符号多项式变换成等价系数向量。

```
>> f=sym('2*x^2+x^3-3*x+5') % f 是字符常量
f =
2*x^2+x^3-3*x+5
>> n=sym2poly(f)
n =
     1     2    -3     5
>> poly2sym(n)
ans =
2*x^2+x^3-3*x+5
>> poly2sym(n,'s') % 's'代替'x'
ans =
s^3+2*s^2-3*s+5
```

2. 符号表达式的代数运算

因为 MATLAB 重载技术的应用, 使得符号表达式的运算符和函数与数值计算基本上相同。但是符号表达式的代数运算与数值运算相比还是有区别:

- ① 符号运算的计算结果比数值运算的结果精确;
- ② 符号运算可以得到一个完全的封闭解或任意精度的数值解;
- ③ 符号运算占用的内存空间比较大且运算时间比较长。

许多标准的代数运算可以在符号表达式上执行, 函数 `symadd`、`symsub`、`symlnul` 和 `symdiv` 为加、减、乘、除两个表达式, `sympow` 将一个表达式上升为另一个表达式的幂次。

【例 3-22】给定以下两个函数。

```
>> f=sym('2*x^2+3*x-5') % 定义符号表达式
f =
2*x^2+3*x-5
>> g=sym('x^2-x+7')
g =
x^2-x+7
>> f+g
ans =
3*x^2+2*x+2
>> f-g
ans =
x^2+4*x-12
>> f*g
ans =
(2*x^2+3*x-5)*(x^2-x+7)
>> f/g
ans =
(2*x^2+3*x-5)/(x^2-x+7)
>> f^3
ans =
(2*x^2+3*x-5)^3
```

3.2 MATLAB符号微积分运算

微分学是微积分的首要组成部分，它的基本概念是导数与微分，其中导数是曲线切线的斜率，返回函数相对于自变量变化的速度；而微分则表明当自变量有微小变化时函数大体上变化多少。积分是微分的逆运算。求给定函数（为导函数）的原函数的运算，这是不定积分——积分学的第一个基本问题。求被积函数在积分上下限区间的计算问题，这是定积分——积分学的第二个基本问题，该问题已由牛顿—莱布尼茨公式解决。微积分学是高等数学重要的基本内容。

3.2.1 符号极限运算

众所周知，微积分中导数的定义是通过极限给出的，即极限概念是数学分析或高等数学最基本的概念，所以极限运算就是微积分运算的前提与基础。函数极限的概念及其运算在高等数学中已经学习过，在此来介绍一下 MATLAB 的符号极限运算的函数命令 `limit()`。函数 `limit()` 调用格式参看以下示例。

【例 3-23】试证明 $\lim_{n \rightarrow 0} (1+n)^{\frac{1}{n}} = e$ ，并求 $\lim_{n \rightarrow a} \frac{\sin x - \sin a}{x - a} = ?$

证： $\lim_{n \rightarrow 0} (1+n)^{\frac{1}{n}} = e$ 其代码如下：

```
>> syms n
limit((1+n)^(1/n),n,0)
```

显示如下：

```
ans = exp(1)
```

即 $\lim_{n \rightarrow 0} (1+n)^{\frac{1}{n}} = e$ 即证成立。

求 $\lim_{n \rightarrow a} \frac{\sin x - \sin a}{x - a} = ?$ 其代码如下：

```
>> syms x a
limit(((sin(x)-sin(a))/(x-a)),a)
```

显示如下：

```
ans = cos(a)
```

即 $\lim_{n \rightarrow a} \frac{\sin x - \sin a}{x - a} = \cos a$ 。

3.2.2 符号函数微分运算

微分运算是高等数学中除极限运算外的最重要的基本内容。

MATLAB 的符号微分运算，实际上是计算函数的导（函）数。MATLAB 系统提供的函数命令 `diff()` 不仅可以求函数的一阶导数，而且还可以计算函数的高阶导数与偏导数。函数的命令 `diff()` 的调用格式参看以下示例。

【例 3-24】已知函数 $f = \begin{bmatrix} \frac{1}{1+x^2} & x e^{x^2} \\ \ln \sin x & x^x \end{bmatrix}$, 试求 $\frac{d^2 f}{dx^2}$ 。

```
>> syms x a;
f=[1/(1+x^2) x*exp(x^2);log(sin(x)) x^x];
dfdx=collect(factor(diff(f,2)))
```

显示如下:

```
dfdx =
[      2*(3*x^2-1)/(1+x^2)^3,      6*x*exp(x^2)+4*x^3*exp(x^2)]
[  -(sin(x)^2+cos(x)^2)/sin(x)^2, x^x*(log(x)^2+2*log(x)+1)+x^x/x]
```

$$\text{即 } \frac{d^2 f}{dx^2} = \begin{bmatrix} \frac{2(3x^2-1)}{(x^2+1)^3} & 2e^{x^2}(3x+2x^3) \\ -\csc^2 x & x^x \left[(\ln x + 1)^2 + \right] \frac{1}{x} \end{bmatrix}$$

【例 3-25】已知函数 $f = \ln(x + \ln y)$, 试求 $\frac{\partial f}{\partial x}$ 与 $\frac{\partial f}{\partial y}$ 。

```
>> syms x y;
f=log(x+log(y));
dfdx=collect(diff(f,x))
dfdy=collect(diff(f,y))
```

显示如下:

```
dfdx = 1/(x+log(y))
dfdy = 1/y/(x+log(y))
```

$$\text{即 } \frac{\partial f}{\partial x} = \frac{1}{(x + \lg y)}, \quad \frac{\partial f}{\partial y} = \frac{1}{y(x + \lg y)}$$

3.2.3 符号函数积分运算

函数的积分是微分的逆运算, 即由已知导(函)数求原函数的过程。函数的积分有不定积分与定积分两种运算。定积分中, 若是积分区间为无穷或被积分区间上有无穷不连续点, 但积分存在或者收敛, 叫做广义积分。MATLAB 系统提供的函数命令 `int()`, 不仅可以计算函数的不定积分, 而且还可以计算函数的定积分以及广义的积分。函数 `int()` 的调用格式参看以下示例。

【例 3-26】已知导数 $\frac{df}{dx} = \begin{bmatrix} e^x & \cos x \\ \sin x & \sqrt{x} \end{bmatrix}$, 试求原函数 $f(x)$ 。

```
>> syms x;
dfdx=[exp(x) sin(x);cos(x) x^(1/2)];
f=int(dfdx)
```

显示如下:

```
f =
[      exp(x),      -cos(x)]
[      sin(x), 2/3*x^(3/2)]
```

$$\text{即 } f(x) = \begin{bmatrix} e^x & -\cos x \\ \sin x & \frac{2}{3}x^{\frac{3}{2}} \end{bmatrix}。$$

【例 3-27】试计算 (1) $\int_0^{\pi} (x \sin x)^2 dx = ?$ (2) $I = \int_a^{3a} dy \int_{y-a}^y (x^2 + y^2) dx = ?$

计算 $\int_0^{\pi} (x \sin x)^2 dx$ 代码如下:

```
>> syms x;
I=int((x*sin(x))^2,x,0,pi)
```

显示如下:

```
I = -1/4*pi+1/6*pi^3
```

即 $\int_0^{\pi} (x \sin x)^2 dx = -1/4\pi + 1/6\pi^3$ 。

计算 $I = \int_a^{3a} dy \int_{y-a}^y (x^2 + y^2) dx$ 代码如下:

```
>> syms a x y;
f=x^2+y^2;
I=int(int(f,x,(y-a),y),y,a,3*a)
```

显示如下:

```
I = 14*a^4
```

即 $I = \int_a^{3a} dy \int_{y-a}^y (x^2 + y^2) dx = 14a^4$ 。

3.2.4 符号求和函数与级数展开函数

无穷级数是高等数学的一个重要组成部分,它是表示函数、研究函数性质以及进行数值近似计算的一种有效工具。高等数学中,无穷级数只介绍常数项级数与函数项级数。级数的收敛性是其重要特征,这是数学分析中讨论的内容。在此主要介绍与级数有关的 MATLAB 符号函数求和与函数展开成 Taylor (泰勒) 级数的问题。

给定一个数列

$$u_1, u_2, u_3, \cdots, u_n, \cdots$$

则由这个数列构成的表达式

$$u_1 + u_2 + u_3 + \cdots + u_n + \cdots$$

叫做常数项无穷级数,记为 $\sum_{n=1}^{\infty} u_n$, 即

$$\sum_{n=1}^{\infty} u_n = u_1 + u_2 + u_3 + \cdots + u_n + \cdots$$

给定一个定义在区间 I 上的函数列 $u_1(x), u_2(x), u_3(x), \dots, u_n(x), \dots$ 则由这个函数列构成的表达式

$$u_1(x) + u_2(x) + u_3(x) + \dots + u_n(x) + \dots$$

叫做定义在区间 I 上的函数项无穷级数。

函数项级数中，简单又常见的一种级数是各项都为幂函数的函数项级数，即所谓幂级数，其形式为

$$a_0 + a_1x + a_2x^2 + \dots + a_nx^n + \dots$$

其中，常数 $a_0, a_1, a_2, \dots, a_n, \dots$ 叫做幂级数的系数。

1. 符号求和函数

收敛的幂级数，不论是常数项级数还是函数项函数，都有求和的问题。在 MATLAB 中提供了级数求和的函数命令 `symsum()`，它的调用格式参看以下示例。

【例 3-28】求幂级数 $\sum_{n=1}^{\infty} \frac{x^{2n-1}}{2n-1}$ 的和。

```
>> syms x n;
f=x^(2*n-1)/(2*n-1);
s=collect(symsum(f,n,1,inf))
```

显示如下：

```
s=1/2*log((1+x)/(1-x))
```

即 $\sum_{n=1}^{\infty} \frac{x^{2n-1}}{2n-1} = 1/2 \log((1+x)/(1-x))$

2. Taylor级数展开函数

给定函数 $f(x)$ ，是否能找到这样一个幂级数，它在某区间内收敛，且其和正好是给定函数 $f(x)$ 。若是能够找到这样的幂级数，则说明函数 $f(x)$ 在该区间内能展开成幂级数。

若函数 $f(x)$ 在点 x_0 的某一邻域内具有从 1 阶到 $(n+1)$ 阶的导数，则在该邻域内，函数 $f(x)$ 在点 $x = x_0$ 时，项数趋向无穷的幂级数如下：

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \dots$$

这个幂级数叫做函数 $f(x)$ 的 Taylor 级数（展开式）。在 MATLAB 系统中，有函数展开成 Taylor 级数的函数命令 `taylor()`，其调用格式参看以下示例。

【例 3-29】试求函数 $f(x) = \sin(x)/x$ 的 Taylor 级数展开式。

```
>> syms x;
f=sin(x)/x;
T=taylor(f,9)
prod2=maple('2!')
prod5=maple('5!')
prod8=maple('8!')
```

显示如下:

```
T =
1-1/6*x^2+1/120*x^4-1/5040*x^6
prod2 =2
prod5 =120
prod8 =40320
```

即 $f(x) = \sin(x)/x$ 的 Taylor 级数展开式为

$$\sin(x)/x = 1 - \frac{x^2}{2!} + \frac{x^4}{5!} - \frac{x^6}{8!}$$

3.3 复变函数运算的MATLAB实现

数从实数拓展到复数, 这是数学发展史中的一个重要里程碑。在实数范围内无法解决 -1 开平方的问题, 在复数范围内就可以迎刃而解。除此之外, 引入复数的概念, 特别是复数引申为向量后, 将正弦时间函数用时间向量来表示, 使复数在电工理论、自动控制理论等自然科学与工程技术中有着广泛的应用。

3.3.1 复数的概念

不需要事先定义 MATLAB 系统定义的数值元素就是复数, 即引入了一个特殊的量 i 或 j , 规定 i 或 $j = \sqrt{-1}$ 。要解方程 $x^2 + 1 = 0$, 其根则为 $x_{1,2} = \pm i$ 。

对于任意两个实数 x 、 y , 称 $z = x + iy$ ($z = x + jy$) 或 $z = x + yi$ ($z = x + yj$) 为复数, 其中 x 、 y 分别叫做复数 z 的实部和虚部, 记作

$$x = \text{Re}(z), \quad y = \text{Im}(z)$$

而 i 或 j 叫做虚单位。当 $x = 0$ 时, $z = iy$ 叫做纯虚数; 当 $y = 0$ 时, 则 $z = x$ 是实数。

两个复数相等, 且它们的实部与虚部必须分别相等。与实数不同, 两个复数不能比较大小。

3.3.2 复变量的函数

1. 关键标识i或j

在 MATLAB 中, 复数单位为 $i=j=\text{sqrt}(-1)$, 其数值在 MATLAB 工作空间中显示为 $0+1.0000i$ 。在 MATLAB 中生成复数, 必须有这个关键标识。

2. 实变量定义标识real

在 MATLAB 中, 通过实变量定义标识 `real` 将构成复变量的两部分设置为实变量后, 再生成标准变量。例如, 将变量 x 与 y 设置为实变量的 MATLAB 语句格式:

```
x=sym('x','real');
y=sym('y','real');
```

也可以用以下的简捷格式:

```
syms x y real;
```

3. 复变量的实部与虚部

求复变量的实部与虚部可用函数命令 `real()` 与 `imag()` 来实现。例如, 有复变量 $z = x + jy$, 求复变量 z 的实部与虚部的 MATLAB 语句如下:

```
>> syms x y real;
z=x+j*y;
Re=real(z)
Im=imag(z)
```

语句执行后就求出复变量 z 的实部与虚部:

```
Re =x
Im =y
```

4. 复变量的模与辐角

求复变量的模与辐角可使用函数命令 `abs()` 与 `angle()`。需要注意的是:

- (1) `angle()` 计算出的辐角的单位是弧度, 且是辐角的主值。
- (2) `angle()` 函数命令只能对数值量进行运算。

【例 3-30】对复变量 $z = x + jy$, 求其 z 的模与辐角。

```
>> syms a th;
x=sym('x','real');
y=sym('y','real');
x=6;y=6;
z=x+j*y;
a=abs(z)
th=angle(z)
th=th*180/pi
```

程序执行后就求出复变量 z 的模与辐角:

```
a =      8.4853
th =      0.7854
th =      45   %辐角
```

5. 共轭复数

将实部相同而虚部同值、正负号相反的两个复数叫做共轭复数。复数的共轭可用函数 `conj()` 来实现, 其调用格式参看以下示例。

【例 3-31】对复数 $z = x - jy$, 求其 z 的共轭复数 \bar{z} 。

```
>> syms x y real;
z=x-y*j;
conj(z)
```

语句执行后就求出复数 $z = x - jy$ 的共轭复数 $\text{ans} = x + jy$

即 $\bar{z} = x + jy$ 。

3.3.3 复数的生成及其矩阵创建

1. 复数的生成

复数可由代数 $z = x + y*j$ (或 $z = x + y*i$) 语句生成, 也可简写成 $z = x + yj$ (或 $z = x + yi$)。复数也可由指数式 (3.3.4 节) $z = A * \exp(j * \text{theta})$ (或 $z = A * \exp(i * \text{theta})$) 语句生成, 也可简写成 $z = A * \exp(\text{theta} * j)$ (或 $z = A * \exp(\text{theta} * i)$), 其中 A 是复数的模, theta 是复数辐角的弧度数。 $z = A * \exp(j * \text{theta})$ 语句执行后, 仍得到复数的代数式。

【例 3-32】用指数式 $z=r*\exp(j*\theta)$ 语句生成复数。

```
>> syms a theta;
a=1;
theta=0.5236;
z=a*exp(theta*j)
```

语句执行生成复数如下：

```
z = 0.8660 + 0.5000i
```

即生成复数 $z=0.866+0.5i$ 。

2. 创建复矩阵

复矩阵就是构成矩阵的元素都是复数或是复变量。创建复矩阵有两种方法，第一种方法与生成一般的矩阵一样，只是将矩阵的元素改换成复变量或复数即可；第二种方法是将实部与虚部矩阵分别创建，然后再结合虚单位 j 写成复数的形式。

【例 3-33】用代数式 $z=x+y*j$ 与指数式 $z=A*\exp(\theta*j)$ 语句生成复矩阵。

```
%用以下 MATLAB 语句生成一个符号复矩阵
>> syms a b c d k f g h z;
z=[-a+b*j c-d*j;k*exp(f*j) -g*exp(h*j)]
```

执行程序显示如下：

```
z =
[ -a+i*b,      c-i*d]
[ k*exp(i*f), -g*exp(i*h)]
```

$$\text{即 } Z_1 = \begin{bmatrix} -a + bj & c - dj \\ ke^{fj} & -ge^{hj} \end{bmatrix}。$$

【例 3-34】用随机函数生成复矩阵。

用 `rand()` 函数生成复矩阵的 MATLAB 语句如下：

```
>> Re=rand(3,2);
Im=rand(3,2);
com=Re+Im*j
```

执行程序显示如下：

```
com =
0.8147 + 0.2785i    0.9134 + 0.9649i
0.9058 + 0.5469i    0.6324 + 0.1576i
0.1270 + 0.9575i    0.0975 + 0.9706i
```

3.3.4 复数的几何意义

任一复数 $z=x+yj$ 与一对有序实数 x, y 一一对应，并且可以用一个平面上某点的两个坐标 (x, y) 来表示，参见图 3-1 所示。这个平面叫做复平面或 z 平面，平面的横轴，即 x 轴称为实轴，纵轴，即 y 轴称为虚轴。这样，复数与复平面上的点形成一一对应的关系。

复数 z 还能用从原点指向点 (x, y) 的向量来表示。向量的长度即为复数的模，几何向量的关系是

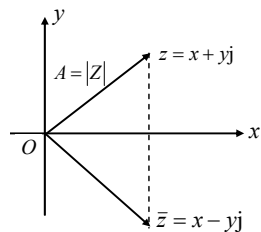


图 3-1 复平面

$$|z| = A = \sqrt{x^2 + y^2}$$

计算向量的长度，在 MATLAB 中正是已经介绍的 `abs()` 函数的功能。

表示复数 z 的向量与 x 轴的夹角 θ 叫做 z 的辐角，记作 $\text{Arg}(z)$ ，其几何关系是

$$\text{Arg}(z) = \theta = \arctan\left(\frac{y}{x}\right)$$

这个关系式用计算向量的辐角，在 MATLAB 中正是已经介绍的 `angle()` 函数的功能。请注意，`angle()` 函数只能对数值量进行运算，如要对符号对象求辐角，只能用上式进行计算。

任何一个复数或向量 $z(z \neq 0)$ 有无穷多个辐角，如果 θ_1 是其中的一个，那么

$$\text{Arg}(z) = \theta_1 = 2k\pi$$

上式给出了 z 的全部辐角，且等式两边角度的单位均为弧度。在 z 的辐角中，把满足 $-\pi < \theta_0 \leq \pi$ 的 θ_0 叫做 $\text{Arg}(z)$ 的主值，记作 $\theta_0 = \text{Arg}(z)$ 。

利用直角坐标与极坐标的关系

$$x = A \cdot \cos(\theta), \quad y = A \cdot \sin(\theta)$$

可有复数 z 的三角表示法

$$z = A \cdot [\cos(\theta) + j\sin(\theta)]$$

有与此对应的复数 z 的代数表示法

$$z = x + yj$$

还可利用 Euler 公式

$$e^{j\theta} = \cos(\theta) + j\sin(\theta)$$

引出复数 z 的指数式表示（或叫极坐标式表示法）

$$z = A \cdot e^{j\theta}$$

复数 z 的指数表示在电工技术与自动控制技术中有着广泛的应用。

【例 3-35】求 4 个复数 $z_1 = 5 + 5j$ ， $z_2 = 5 - 5j$ ， $z_3 = -5 + 5j$ ， $z_4 = -5 - 5j$ 的模与辐角，并写出 4 个复数的指数表示。

```
>> syms z1 z2 z3 z4;
z1=5+5j;z2=5-5j;
z3=-5+5j;z4=-5-5j;
a1=abs(z1)
a2=abs(z2)
a3=abs(z3)
a4=abs(z4)
th1=angle(z1)*180/pi
th2=angle(z2)*180/pi
```

```
th3=angle(z3)*180/pi
th4=angle(z4)*180/pi
```

显示如下：

```
a1 = 7.0711
a2 = 7.0711
a3 = 7.0711
a4 = 7.0711
th1 = 45
th2 = -45
th3 = 135
th4 = -135
```

即 4 个复数的模 $A_1 = A_2 = A_3 = A_4 = 7.0711$ ；还有辐角 $\theta_1 = 45^\circ$ ， $\theta_2 = -45^\circ$ ， $\theta_3 = 135^\circ$ ， $\theta_4 = -135^\circ$

即 4 个复数的指数表示可以直接写出：

$$z_1 = 7.0711 \cdot e^{j45^\circ}, \quad z_2 = 7.0711 \cdot e^{j(-45^\circ)}$$

$$z_3 = 7.0711 \cdot e^{j135^\circ}, \quad z_4 = 7.0711 \cdot e^{j(-135^\circ)}$$

3.3.5 MATLAB在复数代数运算中的实现

两个复数可以进行加法、减法、乘法与除法等代数运算，也可以对复数计算其实部、虚部、模、辐角及共轭复数，复数的加法与减法比较简单，只需要将实部与虚部分别进行加、减就可以了。复数乘法与除法运算要复杂一些。

1. 复数乘法

设有两个复数 $z_1 = A_1 [\cos(\theta_1) + j\sin(\theta_1)]$ ， $z_2 = A_2 [\cos(\theta_2) + j\sin(\theta_2)]$ ，那么

$$\begin{aligned} z_1 \cdot z_2 &= A_1 \cdot A_2 [\cos \theta_1 + j\sin \theta_1] \cdot [\cos \theta_2 + j\sin \theta_2] \\ &= A_1 \cdot A_2 [\cos \theta_1 \cos \theta_2 - \sin \theta_1 \sin \theta_2] + A_1 \cdot A_2 [j(\sin \theta_1 \cos \theta_2 + \cos \theta_1 \sin \theta_2)] \\ &= A_1 \cdot A_2 [\cos(\theta_1 + \theta_2) + j\sin(\theta_1 + \theta_2)] \end{aligned}$$

或者

$$z_1 \cdot z_2 = A_1 \cdot A_2 \cdot e^{j(\theta_1 + \theta_2)}$$

对于 n 个复数有

$$\begin{aligned} z_1 z_2 \cdots z_n &= A_1 A_2 \cdots A_n \cdot [\cos(\theta_1 + \theta_2 + \cdots + \theta_n) + j\sin(\theta_1 + \theta_2 + \cdots + \theta_n)] \\ &= A_1 A_2 \cdots A_n \cdot e^{j(\theta_1 + \theta_2 + \cdots + \theta_n)} \end{aligned}$$

2. 复数除法

设有两个复数 $z_1 = A_1 [\cos(\theta_1) + j\sin(\theta_1)]$ ， $z_2 = A_2 [\cos(\theta_2) + j\sin(\theta_2)]$ ，那么

$$\frac{z_2}{z_1} = \frac{A_2 [\cos \theta_2 + j \sin \theta_2]}{A_1 [\cos \theta_1 + j \sin \theta_1]} = \frac{A_2 e^{j\theta_2}}{A_1 e^{j\theta_1}} = \frac{A_2}{A_1} e^{j(\theta_2 - \theta_1)}$$

【例 3-36】设两个复数 $z_1 = a_1 + jb_1$, $z_2 = a_2 + jb_2$, 试计算 $z_1 + z_2$, $z_1 - z_2$, $\text{conj}(z_1 + z_2)$, $\text{real}(z_1 + z_2)$, $\text{imag}(z_1 + z_2)$, $\text{abs}(z_1)$, $\text{abs}(z_2)$, $\text{abs}(z_1 + z_2)$ 。

```
>> a1=sym('a1','real');
b1=sym('b1','real');
a2=sym('a2','real');
b2=sym('b2','real');
z1=a1+b1*j;
z2=a2+b2*j;
z3=z1+z2
z4=z1-z2
z5=conj(z3)
z6=real(z3)
a7=simple(imag(z3))
a8=abs(z1)
a9=abs(z2)
a10=factor(abs(z3))
```

显示如下:

```
z3 = a1+i*b1+a2+i*b2
z4 = a1+i*b1-a2-i*b2
z5 = a1-i*b1+a2-i*b2
z6 = a1+a2
a7 = b1+b2
a8 =(a1^2+b1^2)^(1/2)
a9 =(a2^2+b2^2)^(1/2)
a10 =(a1^2+2*a1*a2+a2^2+b1^2+2*b1*b2+b2^2)^(1/2)
```

即

$$\begin{aligned} z_1 + z_2 &= a_1 + a_2 + jb_1 + jb_2 \\ z_1 - z_2 &= a_1 - a_2 + jb_1 - jb_2 \\ \text{conj}(z_1 + z_2) &= a_1 + a_2 - jb_1 - jb_2 \\ \text{real}(z_1 + z_2) &= a_1 + a_2 \\ \text{imag}(z_1 + z_2) &= b_1 + b_2 \\ \text{abs}(z_1) &= \sqrt{a_1^2 + b_1^2} \\ \text{abs}(z_2) &= \sqrt{a_2^2 + b_2^2} \\ \text{abs}(z_1 + z_2) &= \sqrt{(a_1 + a_2)^2 + (b_1 + b_2)^2} \end{aligned}$$

【例 3-37】已知两个复数 $z_1 = 5 - 5j$, $z_2 = -3 + 4j$, 试求 $z_1 + z_2$, $z_1 - z_2$, $z_1 \cdot z_2$, z_1 / z_2 , $\text{conj}\left(\frac{z_1}{z_2}\right)$, $\text{abs}\left(\frac{z_1}{z_2}\right)$, $\text{angle}\left(\frac{z_1}{z_2}\right)$ 。

```
>> syms z1 z2 z3 z4 z5 z6 z7 z8;
z1=5-5j;
```

```

z2=-3+4j;
z3=z1+z2
z4=z1-z2
z5=z1*z2
z6=z1/z2
z7=conj(z1/z2)
z8=abs(z1/z2)
th=angle(z1/z2)
th=th*180/pi

```

显示如下：

```

z3 = 2.0000 - 1.0000i
z4 = 8.0000 - 9.0000i
z5 = 5.0000 +35.0000i
z6 = -1.4000 - 0.2000i
z7 = -1.4000 + 0.2000i
z8 = 1.4142
th = -2.9997
th = -171.8699

```

即

$$z_1 + z_2 = 2 - j, \quad z_1 - z_2 = 8 - 9j, \quad z_1 \cdot z_2 = 5 + 35j, \quad z_1 / z_2 = -1.4 - 0.2j$$

$$\operatorname{conj}\left(\frac{z_1}{z_2}\right) = -1.4 + 0.2j, \quad \operatorname{abs}\left(\frac{z_1}{z_2}\right) = 1.4142, \quad \theta = -2.9997, \quad \text{rad} = -171.8699^\circ$$

【例 3-38】试计算 $(1+j)^6$ 与 $(1+j)^{\frac{1}{4}}$ 。

(1) 计算 $(1+j)^6$

```

>> syms z;
z=1+j;
z^6

```

显示如下：

```
ans = 0 - 8.0000i
```

即 $(1+j)^6 = -8j$ 。

(2) 计算 $(1+j)^{\frac{1}{4}}$

```

>> syms z;
z=1+j;
z^(1/4)

```

显示如下：

```
ans = 1.0696 + 0.2127i
```

即 $(1+j)^{\frac{1}{4}} = 1.0696 + 0.2127j$ 。

第 4 章 MATLAB程序设计技术

在 MATLAB 中编写的程序，都保存在 M 文件中。M 文件是统称，每个程序都有自己的 M 文件，文件的扩展名为.m。

MATLAB 有两种 M 文件：脚本和函数。本章的大部分内容是关于函数的，函数比脚本更复杂也更有用。

4.1 MATLAB的控制语句

MATLAB 语言基本的数据元素是数组，而矩阵是数组的特殊形式。因此，过程控制语句中的表达式就可能用到数组和矩阵，这是 MATLAB 与其他语言区别的地方。如果你已经很熟悉其他编程语言中的这些控制语句，在这里只需要掌握矩阵如何使用控制语句就可以了。

4.1.1 条件控制

程序的分支语句，依照条件表达式的值确定程序的走向。

1. if…elseif…else…end语句

表达式为逻辑表达式，只能造成两个分支：要么向左，要么向右。

(1) 一般形式。

```
if logical_expression1
    statements1
elseif logical_expression2
    statements2
else
    statements3
end
```

计算逻辑表达式，根据表达式的值确定是否执行一组语句。

logical_expression1 的值为真，执行 statements1；否则，判断 logical_expression2，为真，执行 statements2。logical_expression1 和 logical_expression2 都不是真，才执行 statements3。

(2) 最简单的形式。

```
if logical_expression
    statements
end
```

在这个基础上，可以增加几个或者一个 elseif，但只能增加一个 else。

if 语句可以嵌套多层，可以嵌套在 if、elseif 或 else 任何语句的下面。

(3) 如果逻辑表达式是非标量值，那么它的所有元素的值都必须不是 0，if 的条件才能满足。假设 x 是一个矩阵，下面的语句是正确的。

```
if x
    statements
end
```

矩阵 x 的全部元素的值都是 1，执行 statements；否则，跳过 statements。
它与下面的语句相同：

```
if all(x(:))
    statements
end
```

(4) if 的逻辑表达式也可以是空数组，表示条件是假。

```
if A
    s1
else
    s0
end
```

A 是一个空数组，因此执行语句 s_0 。

【例 4-1】 包含 else 语句和包含 else if 语句的对比。

<pre>if A x=a else if B x=b else if C x=c else x=d end end end</pre>	<pre>if A x=a elseif B x=b elseif C x=c else x=d end</pre>
----------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------

【例 4-2】 生成一个矩阵。

```
for m=1:k
    for n=1:k
        if m==n
            a(m,n)=2;
        elseif abs(m-n)==2
            a(m,n)=1;
        else
            a(m,n)=0;
        end
    end
end
```

当 $k=6$ 时，得到一个矩阵：

```
a =
    2     0     1     0     0     0
    0     2     0     1     0     0
    1     0     2     0     1     0
    0     1     0     2     0     1
    0     0     1     0     2     0
    0     0     0     1     0     2
```

2. switch...case...otherwise...end语句

可以构造多于两个的分支。程序的走向有多条路，依照条件可以走第一条路，可以走第三条路，可以走第六条路。但是，任你构造多少条路，一次只能选一条。

与 if 语句不同的是，表达式（switch_expr 和 case_expr）是数组或字符串表达式。通过比较 switch_expr 与 case_expr 表达式的值是否相等确定执行哪一组语句。

（1）一般形式。

```
switch switch_expr(numeric expression or string expression)
    case case_expr
        statement,...,statement
    case {case_expr1,case_expr2,case_expr3,...}
        statement,...,statement
    ...
    otherwise
        statement,...,statement
end
```

依据 switch 中变量或表达式的值，执行不同的语句。对于数值表达式（numeric expression），比较表达式与 case 表达式的值；对于字符串表达式（string expression），比较表达式与 case 字符串的值。

（2）switch 能够处理单个 case 中的多个条件，但是，多个条件要放在单元数组中。

```
switch var
    case 1
        disp('1')
    case {2,3,4}
        disp('2 or 3 or 4')
    case 5
        disp('5')
    otherwise
        disp('something else')
end
```

【例 4-3】switch 示例。

```
>> method='Bilinear';
switch lower(method)
    case {'linear','bilinear'}
        disp('Method is linear')
    case 'cubic'
        disp('Mehtod is cubic')
    case 'nearest'
        disp('Mehtod is nearest')
    otherwise
        disp('Unknown method.')
end
```

运行程序结果：

Method is linear

4.1.2 循环控制

程序中总会有对某些量的迭代计算，或者对某个过程的重复处理。将迭代计算和重复处理组织成循环能够简化程序。

有两种循环：次数确定的循环和依条件结束的循环。

1. for...end语句

for 用于指定次数的循环。

(1) 一般形式。

```
for variable=start:increment:n
    statement
    ...
    statement
end
```

variable=start:increment:n 之间使用冒号 (:), 增量 (increment) 在中间, 默认值是 1。

不妨与 VB 做个比较:

```
for Variable=Start To End Step increment
    statements
Next variable
```

经过对比, 很容易记住。

(2) 用数组控制循环次数。假定 A 是 m 行 n 列的数组, for 循环可以写成:

```
for k=A
    statements
end
```

数组的列数作为循环次数。执行时, 把数组的列向量 $A(:, i)$ 赋给 k , 即 $\text{for } k = A(:, i)$, i 为循环次数, 值从 1 到 n 。 k 顺序从列向量中取元素, 每取一个, 执行一次循环。 $k = A(:, 1)$ 是第一次循环, $k = A(:, 2)$ 是第二次循环, 如此下去, 直到 $k = A(:, n)$ 。

(3) 如果被循环执行的语句较少, 整个 for 语句可以写在一行。

例如:

```
for e=eye(4),disp(e),end
```

在 MATLAB 命令窗口运行它:

```
>> for e=eye(4),disp(e),end
    1
    0
    0
    1
```

又如:

```
>> for s=1.0:-0.1:0.0,g=s,k=1/g,x=k*g,end
```



在一行的 for 语句中, 语句之间的分隔符号是逗号。

【例 4-4】for 语句示例。

```
>> a=zeros(k,k) % 全 0 矩阵
for m=1:k
    for n=1:k
        a(m,n)=1/(m+n-1);
    end
end
```

2. while...end语句

依条件结束的循环。

(1) 一般格式。

```
while expression
    statements
end
```

由逻辑表达式 (expression) 控制循环, 表达式是真, 执行 statements; 否则, 退出循环。

(2) 空数组作为表达式。

假如 A 为一空数组:

```
while A
    statements
end
```

(3) 整个 while 写在一行。

```
while A,S1,end
```

如果 A 是一个数组, 只有它的所有元素的值不为 0, 才执行 $S1$; 否则, 不执行 $S1$ 。

【例 4-5】while 语句示例。

```
>> a=[1 0;2 3] % 建立两个矩阵 a,b
b=[2 2;3 3]
while a<b
    c=a(1,1)
end
```

不执行 $c=a(1,1)$, 因为 $a(1,1)$ 不小于 $b(1,1)$ 。

```
>> while b<5
    disp('b 的每个元素都小于 5')
    break
end
```

执行 disp 和 break 语句。

```
while a & b
    disp(b)
end
```

不执行 disp(b), 因为 $a(1,2) \& b(1,2)$ 等于 0 (假)。

3. continue语句

continue 语句用在循环控制中。当你不想执行循环体的全部语句, 只想在做完某一步后直接返回到循环开头时, 在此处插入 continue。continue 后面的语句将被跳过。如果 continue 在嵌套循环中, 它的作用只在封闭它的最内层循环。

【例 4-6】包含 continue 语句的循环。

```
>> fid=fopen('magic.m','r');
count=0;
while ~feof(fid)
    line=fgetl(fid);
    if isempty(line)| strcmp(line,'% ',1)
        continue
    end
    count=count+1;
end
disp(sprintf('%d lines',count));
```

4. break 语句

break 语句用在 for 和 while 中，立即结束循环，而继续执行循环之外的下一条语句。在嵌套循环中，break 只使程序从封闭它的最内层循环退出。

【例 4-7】包含 break 语句的循环。

```
fid=fopen('fft.m','r');
a="";
while ~feof(fid)
    line=fgetl(fid);
    if isempty(line)
        break
    end
    a=strvcat(a,line);
end
disp(a)
```

4.1.3 错误控制

错误控制在程序中是不可缺少的，它是对程序出错的预防和治疗，不会使程序误入歧途。不会造出莫名其妙的结果。

try...catch...end 语句

它提供了一种检测和处理错误的方法。当执行中的程序在 try...catch 的范围内出现错误时，它能捕捉到错误，并能立即转到处理错误的代码段 catch...end，对错误进行处理或做出响应，也就是给错误一个出口，以避免编程人的迷乱。

一般形式：

```
try
    statement
    ...
    statement
catch
    statement
    ...
    statement
end
```

如果程序在 `catch...end` 之间出错, MATLAB 结束程序的执行。除非在 `catch...end` 中有 `try...catch` 代码段。

【例 4-8】`try` 语句示例。

```
function matrixMultiply(a,b)
try
    x=a*b
catch
    disp '** Error multiplying a*b'
end
```

在 MATLAB 命令窗口输入两个矩阵, 然后运行上面的函数。

```
>> a=[1 4 7;2 5 8;3 6 9];
>> b=[1 5 9;3 5 7];
>> matrixMultiply(a,b)
```

显示如下:

```
** Error multiplying a*b
```

显然错误出在 $x = a * b$ 。为什么错了? 因为 a 的列数不等于 b 的行数。



例子是一个函数, 它不能在 MATLAB 的命令窗口写入, 应该把它写在 M 文件中, 然后从命令窗口调用。

4.2 M文件编程

4.2.1 M文件的分类介绍

M 文件分为两种, 一种是脚本文件 (Scripts File), 另一种是函数文件 (Function File)。两种文件的比较如下。

MATLAB 脚本文件特点如下:

- 通常为一连串指令。
- 无输入参数和返回参数。
- 利用的数据和产生的中间结果都保存在 MATLAB 的基本工作空间。

MATLAB 函数文件特点如下:

- 在扩充 MATLAB 函数库中使用。
- 可以接收参数和返回参数。
- 利用的数据和产生的中间结果都保存在函数本身独立的工作空间中。

由上面的比较可以看出, 脚本文件的数据由于保存在 MATLAB 基本工作空间中, 所以, 容易与其他操作产生变量混淆。比如, 某脚本文件中有下面的语句:

```
average=sum/length;
```

而 MATLAB 工作空间中已经存在一个 `average` 变量, 这样, 调用此脚本文件后, 原 `average` 变量就被覆盖了。后面利用到 `average` 的操作会以原来的数值进行运算, 从而产生难以察觉

的错误。所以，利用脚本文件时，除非需要，变量名字力求不和 MATLAB 工作环境中的变量重复。

1. M脚本文件

MATLAB 指令类似于 DOS 命令，而脚本文件类似于 DOS 系统中的 .bat 批处理文件。即脚本文件是一连串 MATLAB 指令，可以将烦琐的计算或操作放在一个 M 文件里面，每当调用这一连串指令时，只需要输入 M 文件名即可，从而简化了操作。

脚本与 MATLAB 会话共享基本工作空间。它们主要是操作工作空间中的数据，也可以在工作空间中产生新的数据，继续进行下一步的运算。

【例 4-9】编写脚本文件，根据不同的 θ 用三角函数计算 ρ 多次，然后根据 θ 和 ρ 的值画图。

```
%定义 example4_1.m 文件
%An M-file script to produce
%"flower petal" plots
theta=-pi:0.01:pi;
rho(1,:)=2*sin(5*theta).^2;
rho(2,:)=cos(10*theta).^3;
rho(3,:)=sin(theta).^2;
rho(4,:)=5*cos(3.5*theta).^3;
for k=1:4
    polar(theta,rho(k,:)) %图形输出
    pause
end
```

在编辑器上输入以上程序，文件保存为 example4_1.m，这时 example4_1.m 就是一个 MATLAB 脚本。在控制窗口的命令行中输入 example4_1 运行此脚本，运行结果 4-1 所示。

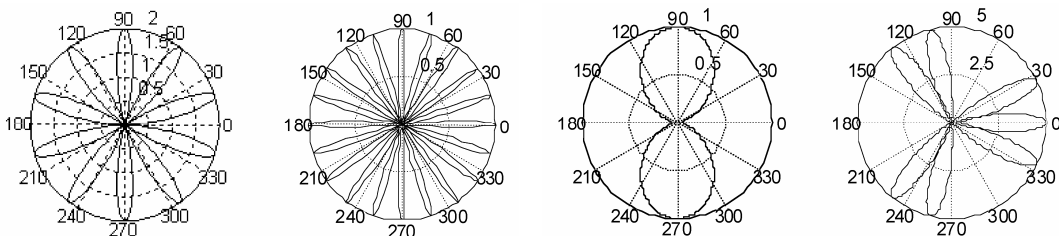


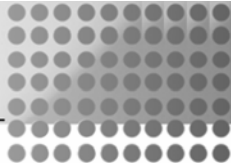
图 4-1 example4_1 的执行结果

由于使用 pause 指令，4 张图片没有同时显示，按回车键，可以让另外 3 张图片陆续显示出来。此程序没有输入参数和输出参数，但是程序创建了必要的变量，如 θ 和 ρ ，脚本运行结束后，这些变量会自动保存在 MATLAB 基本工作空间中。

2. M函数简介

与脚本文件不同，函数文件可以接受输入变量，并可返回结果。M 函数文件通常在扩充 MATLAB 函数库中使用，并且可以接受参数，也可以返回参数，而且不像 C 语言一个函数只能返回一个值，MATLAB 函数可以返回任意多个值。

M 函数文件的实现对于用户来说是透明的。M 函数文件运行时，会创建此函数的函数工作空间 (Function Workspace)，运算中产生的变量都存放在这个工作空间中，与其他函数空间和 MATLAB 基本工作空间相独立。因此大型的程序宜用函数文件，便于封闭与调用。



【例 4-10】计算一个向量所有元素的平均值。

```
%定义 aver.m 函数

function y=aver(x)
% 计算向量元素的平均值
% aver(x)，为一个向量 x 元素的平均值
% 如果没有输入向量，程序将报错
[m,n]=size(x);
if(~((m==1)|(n==1))|(m==1 & n==1))
    error('please input a vector')
end
y=sum(x)/length(x) %计算
```

这个例子包含了典型的 M 函数的各个部分：函数定义行、H1 行、帮助文档、函数主体以及注释。

函数编辑完成后，将文件保存为 `aver.m`。程序中要求有一个输入参数，在命令行中输入 `z` 向量并赋值：

```
>> z=1:199;
z=1:199 是定义函数的输入参数，输入文件名调用此函数：
>> aver(z)
```

输出结果为：

```
ans =    100
```



函数类型将在下一节展开介绍，在此先介绍函数调用和变量传递。

4.2.2 函数调用和变量传递

MATLAB 中函数的调用方法与 C 语言中的调用方法相似，可以在控制窗口以命令行形式调用，也可以在 M 函数中调用。而 MATLAB 的参数传递一般是值传递，另外，MATLAB 支持多个返回参数。

1. 函数调用

当从命令行或 M 文件调用另外一个 M 文件时，MATLAB 把函数解析成伪码 (Pseudocode) 并存储在内存中。这样在下次调用此函数时，就不必再次对此函数解析了。除非用 `clear` 函数清除或退出 MATLAB，否则，此伪码一直会驻留在内存。

- `clear functionname`：清除指定文件名的伪码。
- `clear functions`：清除所有 M 函数的伪码。
- `clear all`：清除内存中所有变量和函数。

(1) 函数调用顺序

当多个函数有相同的函数名时，MATLAB 将根据函数的位置及类型按顺序调用。MATLAB 调用函数的顺序遵循以下原则。

① 变量。在进行函数名匹配之前，MATLAB 先在当前工作空间查找是否存在以此命名的变量。若存在，则认为是调用此变量同时结束查找。



- ② 子函数。子函数比相同路径上其他的 M 函数和重载函数的优先级要高。
- ③ 私有函数。
- ④ 类的构造函数。
- ⑤ 重载函数。
- ⑥ 当前路径上的函数。
- ⑦ 搜索其他路径。

第③条到第⑦条中的函数都可以是下面 5 种类型之一：M 函数、内嵌函数、P 伪码、MEX 文件和 Simulink model（MDL 文件）。对于这 5 种类型的函数，也有其如下调用顺序。

- ① MATLAB 的内嵌函数。
- ② MEX 文件。
- ③ MDL 文件。
- ④ P 伪码文件。
- ⑤ 用户的 M 文件。

用 which 命令可以查询 MATLAB 会调用哪个函数。比如：

```
>> which pie3
D:\MATLABR2008\toolbox\matlab\specgraph\pie3.m
```

（2）函数调用语法

命令行方式调用函数的语法如下，函数名后跟参数列表，函数名和参数以及各参数之间用空格符隔开：

```
functionname in1 in2...inN
```

命令行的函数语法比较简单，容易写，但是，缺点是不能为函数的返回参数赋值。

命令行调用函数的简单示例如下：

```
save mydata.mat x y z
clear length width depth
```

函数式的语法和其他编辑语言相似，MATLAB 的特别之处在于一个函数可以返回多于一个的参数。

只有一个返回值的函数语法如下：

```
out=functionname(in1,in2,...,inN)
```

若函数返回多于一个参数，参数之间用逗号或空格隔开，然后用方括号（[]）把所有参数括起来：

```
[out1,out2,...,outN]=functionname(in1,in2,...,inN)
```

两个简单示例如下：

```
>> copyfile(srcfile,'..\mytests','wriable')
[x1,x2,x3,x4]=deal(A{:})
```

用函数式调用函数时，对参数表中的参数进行赋值，例如，下面的表达式为 polyeig 函数的参数 A0、A1 和 A2 赋值：

```
>> e=polyeig(A0,A1,A2)
```

用命令行式调用函数，参数以文字字符的形式传递，下面的表达式为 save 函数传递了字符串形式的参数 mydata.mat、x、y 和 z：

```
save mydata.mat x y z
```

【例 4-11】以例子来说明两种传递参数方式的不同。

```
%先函数式调用 disp 函数
A=pi;
disp(A)
%显示结果为 pi 的值: 3.1416
%再以命令行形式调用 disp A,传递字符串参数'A'
A=pi;
disp A
%显示结果为字符'A': A
```

以命令行执行 disp A 时，会把 A 当做字符串，而不是变量，所以会显示 A，而非 A 所代表的内容。

【例 4-12】用命令行形式和函数式分别为 strcmp 函数传递两个有相同内容的字符串，查看结果。

```
>> str1='one';
str2='one';
strcmp(str1,str2)
```

显示如下：

```
ans = 1    （相等）
```

用函数式调用 disp 函数时，得到的结果为 1，表示两字符串相等（相同）。

```
str1='one';
str2='one';
strcmp str1 str2
```

显示如下：

```
ans = 0    （不相等）
```

用命令行方式调用 disp 函数时，得到的结果为 0，即不相等（相同），因为在这种情况下，由于命令行式调用函数，参数是以字符串形式传递的，所以，strcmp 函数比较的是字符串 str1 和 str2，而非 str1 和 str2 的值 one。

2. 参数传递

对函数进行调用时，返回参数个数可以少于函数定义时的返回参数个数，但是不可以多于。比如，一个函数定义有 N 个返回参数，但是调用时，可以使用 0 到 N 个返回参数。不需要的返回参数被丢弃。函数调用时，按照函数定义行指定的顺序来返回参数。

【例 4-13】举例说明参数返回的顺序。

```
function [a b c]=myfun(x,y)
b=x*y;
a=99;
c=x.^2;
```


上例先返回 99, 然后是 $x*y$, 最后是 $x.^2$, 虽然先计算得到了 $x*y$, 但是参数列表中 a 在 b 前面, 故先返回 a 的值。

当返回参数个数少于函数定义的返回参数个数时, 尤其要注意上面的顺序。如调用时一个返回参数:

```
>> a=myfun(x,y)
```

此时仅仅返回 99 而不是 $x*y$ 的值, b 和 c 的值被丢弃。如调用时无任何返回参数:

```
myfun(x,y)
ans= 100
```

(1) 检查参数个数

与 C 语言一样, MATLAB 可获取输入参数个数信息并根据不同输入参数个数完成不同的功能。MATLAB 中用 `nargin` 和 `nargout` 函数获取函数调用时输入参数和输出参数的个数。

【例 4-14】对输入参数判断, 然后完成不同的任务。

```
function c=testarg(a,b)
if(nargin==1)
    c=a.^2;
elseif(nargin==2)
    c=a+b;
end
```

(2) 可选输入/输出参数

利用 `varargin` 函数和 `varargout` 函数可以传递任意数目的输入参数和输出参数。MATLAB 将所有的输入参数打包成一个细胞数组, 而输出参数需要自己编写代码打包成细胞数组以便 MATLAB 将输出参数传递给调用者。

使用 `varargout` 来返回可选的参数值有下面两种定义形式:

```
function varargout=myfun(vin1,vin2,...)
function [vout1 vout2...varargout]=myfun(vin1,vin2,...)
```

用第一种定义形式定义的函数, 其函数体内创建了 `varargout` 细胞数组。此细胞数组的元素及其顺序决定了函数被调用时 MATLAB 如何为可选的返回值赋值。这种情况下 `varargout` 是函数定义行中等号左边唯一的变量, MATLAB 将 `varargout{1}` 赋值给第一个返回参数, 将 `varargout{2}` 赋值给第二个返回参数, 依次类推。

用第二种定义形式定义的函数, 除了 `varargout`, 函数定义行中还有其他的返回参数。此时 MATLAB 先为调用函数最左边返回的参数赋值, 然后按照顺序为 `varargout` 数组赋值。

【例 4-15】倒序逐行输出 5 阶魔方矩阵的值。

```
function varargout=byRow(a)
varargout{1}='With VARARGOUT constructed by row...';
for k=1:5
    row=5-(k-1); %翻转行的顺序
    varargout{k+1}=a(row,:);
end
```

`varargout{1}` 赋值为字符串常量。指定 4 个返回变量调用此函数, 则 MATLAB 返回 `varargout{1:4}`, 其中, `varargout{2:4}` 返回矩阵的倒数三行。结果如下:

```
>> [text r1 r2 r3]=byRow(magic(5))
text =
With VARARGOUT constructed by row...
r1 =
    11    18    25     2     9
r2 =
    10    12    19    21     3
r3 =
     4     6    13    20    22
```

byRow 函数可以用 0~6 个返回参数来调用，如下面的调用方式都是正确的：

```
>> [text r1 r2 r3 r4 r5]=byRow(magic(5)) %6 个返回参数
byRow(magic(5)) %不用返回参数
```

显示如下：

```
text =
With VARARGOUT constructed by row...
r1 =
    11    18    25     2     9
r2 =
    10    12    19    21     3
r3 =
     4     6    13    20    22
r4 =
    23     5     7    14    16
r5 =
    17    24     1     8    15
ans =
With VARARGOUT constructed by row...
```

【例 4-16】说明使用 varargin，下面的函数可以接受任意多个二维向量，然后用直线将以这些二维向量为坐标的点连接起来。

```
function testvar(varargin)
for k=1:length(varargin)
    x(k)=varargin{k}(1); %细胞矩阵索引
    y(k)=varargin{k}(2);
end
xmin=min(0,min(x));
ymin=min(0,min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

testvar 函数可以接受任意多的参数，比如，下面调用方式都是可以的：

```
testvar([2 3],[1 5],[4 8],[6 5],[4 2],[2 3])
testvar([-1 0],[3,-5],[4,2],[1 1])
```

由于 varargin 用一个细胞数组包含了所有的输入参数，所以可以用细胞数组的索引来获取每个参数的值。例如：

```
>> y(n)=varargin{n}(2);
```

细胞数组索引有以下两个部分，用花括号{}的索引表示细胞数组细胞的索引，用圆括号()的索引表示某细胞的内容索引。像上面的代码，表达式{n}访问 varargin 细胞数值的第 n 个值，表达式(2)访问第 n 个细胞的第二个内容。

【例 4-17】演示使用 varargin，输入参数的矩阵必须是 2 列，而行数可以任意，第 1 列为 x 坐标集合，第 2 列为 y 坐标集合。

```
function [varargout]=testvar2(arrayin)
for k=1:nargout
    varargout{k}=arrayin(k,:); %细胞数组赋值
end
```

函数把每一行的坐标对分离成单独的[xi yi]，这些坐标值组成一个坐标向量。

调用 testvar2:

```
>> a=[1 5;3 7;2 8;4 6;2 7];
[p1,p2,p3,p4,p5]=testvar2(a)
p1 =
     1     5
p2 =
     3     7
p3 =
     2     8
p4 =
     4     6
p5 =
     2     7
```

varargin 和 varargout 必须出现在参数列表的最后，前面可以是任意个输入或输出参数。下面的 varargin 和 varargout 位置都是正确的：

```
function [out1,out2]=example1(a,b,varargin)
function [i,j,varargout]=example2(x1,y1,x2,y2,flag)
```

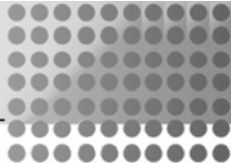
在嵌套函数中使用可选参数时要特别注意，因为在嵌套函数中的 varargin, varargout, nargin 和 nargout 的具体含义可能会发生混乱。varargin 和 varargout 是变量，和其他变量一样，遵循 MATLAB 的变量作用域原则。另外，注意嵌套函数与所有外部函数共享工作空间。若 nargin 或 nargout 出现在嵌套函数中，则代表传递给这个函数的输入参数或输出参数个数，不管这个函数是否为嵌套函数。若嵌套函数需要得到外部函数的 nargin 值和 nargout 值，可以将此作为参数传递。nargin 和 nargout 为函数，会记录被调用时的输入参数和输出参数。

若函数体中对输入参数进行修改，则需要将此参数也列入输出参数，这样调用函数就能得到修改后的值。例如：

```
function [text,offset]=readText(filestart, offset)
```

调用 readText 函数时，读取某文件一行，则必须记录此次文件的偏移量 offset 以便下次调用时能获取开始读取的位置。但是每次 readText 函数调用结束后，offset 变量的值就从内存中清除了。为保存 offset 的值，采用上述调用方法，将 offset 作为返回值。

另外，MATLAB 提供了一个 inputParser 类来处理传递给 M 文件函数中不同类型的参数。



4.2.3 数据导入与导出

MATLAB 提供了将磁盘文件或剪贴板中的数据加载到工作空间的多种方法,称之为导入数据 (Importing Data),同时也提供了多种将工作空间的变量保存到磁盘的方法,称之为导出数据 (Exporting Data)。

选择不同的导入机制或导出机制取决于要传输的数据的格式,比如,文本文件、二进制文件和 JPEG 文件。MATLAB 内嵌入了以下导入与导出文件。

- 二进制文件。
- 文本文件。
- 图形文件。
- 音频或视频文件。
- 电子数据表 (Spreadsheets)。
- 剪贴板的数据。
- Internet 的信息。

除了 MATLAB 的导入函数外,还可以用工具箱来导入具有特定特点的数据,比如,可以使用 Database Toolbox 来导入关系数据库的数据。

1. 使用导入向导

导入向导是 MATLAB 提供的一个图形交互界面,大大方便了数据的导入。若从文件导入数据,则单击【File】菜单下的【Import Data...】命令或在命令行打开导入数据向导:

```
>> uiimport -file
```

若从剪贴板导入数据则单击【Edit】菜单下的【Paste to WorkSpace】命令,或输入下面的命令行:

```
>> uiimport -pastespecial
```

【例 4-18】导入一个文本文件的数据到 MATLAB 工作空间。

文本文件 grad.txt 的内容如下:

```
John 85 90 95
Ann 90 92 98
Martin 100 95 97
Rob 77 86 92
```

打开导入向导对话框导入 grad.txt,如图 4-2 所示。

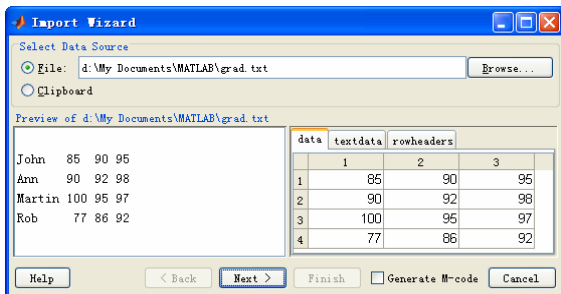


图 4-2 “ImportWizard”对话框



界面的左边为文件内容预览，以文本格式显示，和用其他文本编辑器打开显示的内容与格式是相同的。界面的右边有 3 项内容：**data**、**textdata** 和 **rowheaders/colheaders**。其中，**data** 为文件中的数据，指数字部分；**textdata** 为文件中除数字部分外的数据，因为 MATLAB 默认导入数字数据；**rowheader/colheader** 是行名或列名，如本例中的第 1 列为每一行的行名，MATLAB 将其归为 **textdata** 中。可以选择是否需要导入 **textdata** 和 **rowheaders/colheaders**，可以选择将数据的每一行以行名为变量名创建一个单独的行向量后导入，也可以仅导入用户指定的行。

在控制窗口使用下面的命令也可以导入 **grad.txt** 的数据：

```
>> uiimport grad.txt %grad.txt 要在当前目录下
>> whos
```

Name	Size	Bytes	Class	Attributes
data	4x3	96	double	
rowheaders	4x1	272	double	
textdata	4x1	272	double	

2. 导入导出MAT文件

使用 **save** 函数可以将工作空间的变量导出为二进制或 ASCII 文件。可以保存工作空间中的所有变量或保存指定的某些变量。

将所有变量保存到 **filename** 文件中：

```
save filename
```

保存指定的变量：

```
save filename var1 var2...varN
```

在指定保存所需变量时，变量名称中可包含通配符“*”，下面的命令保存所有开头为 **str** 的变量：

```
save strinfo str*
```

用 **whos -file strinfo** 命令可以检查导入到此 MAT 文件的数据：

```
>> whos -file strinfo
```

Name	Size	Bytes	Class	Attributes
str2	1x15	30	char array	
strarray	2x5	678	cell array	
strlen	1x1	8	double array	

保存 MATLAB 结构体变量时，可以将结构体作为一个整体变量保存，也可以将结构体的每个域作为单独的变量保存，还可以以单独变量的形式保存某些指定的域。比如，对于结构体 **S**。

```
S.a=12.7; S.b={'abs',[4 5;6 7]}; S.c='Hello!';
```

保存整个结构体：

```
>> save newstruct.mat S;
whos -file newstruct
```

Name	Size	Bytes	Class
S	1x1	550	struct array

使用-struct 选项单独保存每个域为独立的变量:

```
save newstruct.mat -struct S;
```

```
whos -file newstruct
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x2	158	cell array
c	1x6	12	char array

或保存指定的域为独立的变量:

```
save newstruct.mat -struct S a c;
```

```
whos -file newstruct
```

Name	Size	Bytes	Class
a	1x1	8	double array
c	1x6	12	char array

load 函数可将磁盘上的二进制文件或 ASCII 文件导入到 MATLAB 工作空间:

```
load filename
```

或导入指定的变量 (同样也可以使用通配符 “*”):

```
load filename var1 var2...varN
```

也可以将 MAT 文件中的函数导入到一个结构体中:

```
S=load('mydata.mat')
```

3. 导入导出图形文件

使用 imread 函数可以将图形文件导入到 MATLAB 工作空间。imread 函数支持标准文件格式的图形文件, 包括 TIFF (Tagged Image File Format)、GIF (Graphics Interchange Format)、JPEG (Joint Photographic Experts Group) 以及 PNG (Portable Network Graphics) 格式。下面的命令将 JPEG 格式的图形数据读取到 MATLAB 工作空间, 用数组 I 存储:

```
>> I=imread('moon.jpg');
```

imread 用多维数组来表示图像数据, 维数取决于图像的格式。比如, 使用三维数组代表 RGB 颜色图像:

```
>> whos I
```

Name	Size	Bytes	Class	Attributes
I	120x160x3	57600	uint8	

imwrite 函数可从 MATLAB 工作空间中导出标准格式的图形文件, 支持的格式与 imread 相同。下面的命令将 MATLAB 工作空间中的多维数组数据 I 读取到 TIFF 格式的文件中:

```
imwrite(I,'my_ph.tif','tif');
```

4. 导入导出音频视频文件

MATLAB 有很多函数可以查询包含音频或视频数据的文件信息, 如 mmfileinfo 函数。同时, MATLAB 提供了很多导入音频视频数据到工作空间的函数, 可以从文件中导入, 也可以利用输入设备录制, 如用麦克风。

导入音频视频的函数有 auread、aviread 和 wavread, 可分别读取声音文件、AVI 视频和 WAVE 声音。

MATLAB 中导出音频的函数有 `auwrite` 和 `wavwrite`, 分别可导出声音为 AU 和 WAV 格式的文件。而导出视频文件就复杂一些, 需要用 `avifile` 函数创建 `avifile` 对象, 然后利用 AVI 文件对象的方法和属性来控制导出过程。如在 MATLAB 中, 可把一系列的图形保存为可播放的电影, 然后导出为 MAT 文件。

【例 4-19】创建 AVI 文件: 将一系列图像保存为 AVI 格式的电影。

```
% 1 创建 AVI 文件对象
aviobj=avifile('mymovie.avi','fps',5);
%2 抓图, 存进 AVI 文件中
for k=1:25
    h=plot(fft(eye(k+15)));
    set(h,'EraserMode','xor');
    axis squal
    frame=getframe(gca);
    aviobj=addframe(aviobj,frame);
end
%3 关闭 AVI 文件
aviobj=close(aviobj)
```

5. 导入导出电子数据表

MATLAB 支持微软的 Excel 电子数据表和 Lotus 的 1-2-3 电子数据表。由于前者用户较多, 在此只介绍 Excel 的导入导出。

使用 `xlswrite` 函数可将矩阵导出为 Excel 表。例如包含文字和数字混合数据的矩阵:

```
>> d={'Time','Temp';13 100;15 99;18 96}
d =
    'Time'    'Temp'
    [ 13]    [ 100]
    [ 15]    [  99]
    [ 18]    [  96]
```

导出为名为 `tempdata.xls` 的 XLS 文件, 标签名为 `Temperatures`, 数据从 E1 位置开始写:

```
>> xlswrite('tempdata.xls',d,'Temperatures','E1')
```

`xlsread` 函数可将 Excel 文件的数据导入到 MATLAB 工作空间。例如, 将上面导入的 XLS 文件中的数据导入:

```
>> ndata=xlsread('tempdata.xls','Temperatures')
ndata =
    13    100
    15     99
    18     96
```

MATLAB 仅仅导入数字信息, 而忽略了文字信息。若要将文字数据也导入, 需要用两个返回参数:

```
>> [ndata,headertext]=xlsread('tempdata.xls','Temperatures')
headertext =
    'Time'    'Temp'
```

```
ndata =
    13    100
    15     99
    18     96
```

4.2.4 示例分析

【例 4-20】分别建立脚本文件和函数文件，将华氏温度 F 转换为摄氏温度 C。

%程序 1：首先建立脚本文件并命名为 example20_A.m 存储

```
f=input('Input Fahrenheit temperature:');
c=5*(f-32)/9;
```

然后在 MATLAB 命令窗口中输入 example20_A，将执行该脚本文件，执行情况如下：

```
>> example20_A
Input Fahrenheit temperature:84
c =
    28.8889
%程序 2：首先建立函数文件 example20_B.m
function c=example20_B(f)
c=5*(f-32)/9;
```

然后在 MATLAB 命令窗口调用该函数文件：

```
y=input('Input Fahrenheit temperature:');
x=example20_B(y);
```

输出情况如下：

```
x =
    28.8889
```

【例 4-21】输入 x 、 y 的值，并将它们的值互换后输出。

```
%用 input 函数进行编写
x=input('Input x please. ');
y=input('Input y please. ');
z=x;
x=y;
y=z;
disp(x);
disp(y);
```

【例 4-22】求一元二次方程 $ax^2 + bx + c = 0$ 的根。

```
%用 input 获得系数参数的值
a=input('a=');
b=input('b=');
c=input('c=');
d=b*b-4*a*c;
%用求根公式求值
x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)];
disp(['x1=',num2str(x(1))','x2=',num2str(x(2))]);
```


【例 4-23】计算分段函数的值 y 。

$$\text{分段函数为} \begin{cases} \frac{x + \sqrt{x}}{e + x^2}, & x \leq 0 \\ \lg(x + \sqrt{1 + x^2}) / 2, & x > 0 \end{cases}$$

代码如下：

```
c=x=input('请输入 x 的值: ');
if x<=0
    y=(x+sqrt(pi))/(e+x^2)
else
    y=lg(x+sqrt(1+x^2))/2
end
```

【例 4-24】某商场对顾客所购买的商品实行打折销售，标准如下（商品价格用 p 来表示）。

$p < 100$	没有折扣
$100 \leq p < 500$	4%折扣
$500 \leq p < 1000$	6%折扣
$1000 \leq p < 1600$	8%折扣
$1600 \leq p < 2500$	12%折扣
$2500 \leq p$	15%折扣

输入所售商品的价格，求其实际销售价格。

```
%用 switch 语句实现
p=input('请输入商品价格')
switch fix(p/100)
    case{0,1} %价格小于 100
        rate=0;
    case{2,3,4} %价格大于等于 100 但小于 500
        rate=4/100;
    case num2cell(5:9) %价格大于等于 500 但小于 1000
        rate=6/100;
    case num2cell(10:15) %价格大于等于 1000 但小于 1600
        rate=8/100;
    case num2cell(16:25) %价格大于等于 1600 但小于 2500
        rate=12/100;
    otherwise %价格大于等于 2500
        rate=15/100;
end
p=p*(1-rate) %输出商品实际销售价格
```

【例 4-25】输入一个字符，若为大写字母，则输出其对应的小写字母；若为小写字母，则输出其对应的大写字母；若为数字字符，则输出其对应的数值，若为其他字符，则原样输出。

```
%用 if-else 语句实现
c=input('请输入一个字符:','s');
if c>='A' & c<='Z'
    disp(setstr(abs(c)+abs('a')-abs('A')));
```

```
elseif c>='a' & c<='z'
    disp(setstr(abs(c)-abs('a')+abs('A')));
elseif c>='0' & c<='9'
    disp(abs('c')-abs('0'));
else
    disp(c);
end
```

【例 4-26】矩阵乘法运算，要求两矩阵的维数相容，否则会出错。先求两矩阵的乘积，若出错，则自动转去求两矩阵的点乘。

```
%用 try-catch 语句实现
a=[1 5 9;3 5 7];
b=[11 21 31;12 22 32];
try
    c=a*b;
catch
    c=a.*b;
end
lasterr      %显示出错原因
```

运行此脚本，显示如下：

```
ans =
Error using ==> mtimes
Inner matrix dimensions must agree.
```

【例 4-27】已知 $y = 1 + \frac{1}{3} + \frac{1}{5} + \cdots + \frac{1}{2n-1}$ ，当 $n = 100$ 时，求 y 的值。

代码如下：

```
>> y=0;
n=100;
for i=1:n
    y=y+1/(2*i-1);
end
y
```

输出如下：

```
y =      3.2843
```

在实际 MATLAB 编辑中，采用循环语句会降低其执行速度，所以前面的程序通常由下面的程序来代替：

```
n=100;
i=1:2:2*n-1;
y=sum(1./i);
```

【例 4-28】一个三位整数，各位数字的立方和等于该数本身，则称该数为水仙花数。输出全部水仙花数。

代码如下：

```

for m=100:999
    a1=fix(m/100); %求 a 的百位数字
    a2=rem(fix(m/10),10); %求 a 的十位数字
    a3=rem(m,10); %求 a 的个位数字
    if a==a1*a1*a1+a2*a2+a2+a3*a3*a3
        disp(a)
    end
end
end

```

【例 4-29】写出下列程序的执行结果。

```

s=0;
a=[11,21,31;12,22,32;13,23,33;14,24,34];
for k=a
    s=s+k;
end
disp(s');
for 语句更一般的格式如下:
for 循环变量=矩阵表达式
    循环体语句
end

```

执行过程是依次将矩阵的各列元素赋给循环变量，然后执行循环体语句，直至各列元素处理完毕。故上述代码中 for 循环中即是将矩阵的各行分别累加，最后结果为列向量。

本例执行结果如下：

```

63    66    69    72

```

【例 4-30】求任意两个数的最大公约数和最小公倍数。

```

A1=input('输入两个非零数 第一个:');
A2=input('输入两个非零数 第二个:');
a=max(A1,A2);
b=min(A1,A2);
while(b~=0)
    r=rem(a,b);
    a=b; %利用代数学中的辗转相除法
    b=r;
end
disp(a);
disp(A1*A2/a); %A, B 两数的最小公倍数为 A×B/(A 和 B 的最大公约数)

```

完成一个功能，选择正确高效率的算法是关键。

【例 4-31】从键盘输入若干个数，当输入 0 时结束输入，求这些数的平均值和它们之和。

```

%用 while 语句实现，结束条件为输入字符'0'
sum=0;
cnt=0;
val=input('Enter a number(end in 0):');
while(val~=0)
    sum=sum+val;

```

```
    cnt=cnt+1;
    val=input('Enter a number(end in 0):');
end
if(cnt>0)
    sum
    mean=sum/cnt
end
```

【例 4-32】若一个数等于它的各个真因子之和，则称该数为完数，如 $6=1+2+3$ ，所以 6 是完数。求[1, 499]之间的全部完数。

```
for m=1:499
    s=0;
    for k=1:m/2
        if rem(m,k)==0
            s=s+k;
        end
    end
    if m==s
        disp(m);
    end
end
```

显示如下：

```
6
28
496
```

【例 4-33】求[100, 200]之间第一个能够被 21 整除的整数。

```
for n=100:200
    if rem(n,21)~=0
        continue
    end
    n
    break
end
```

显示如下：

```
n =    105
```

【例 4-34】鸡兔同笼问题：鸡和兔关在一个笼子里，已知共有 36 个头，100 只脚，求笼内关了多少只兔子和多少只鸡。

```
for chicken=1:36
    if rem(100-chicken*2,4)==0 & (chicken+(100-chicken*2)/4)==36
        break
    end
    chicken=chicken+1;
end
```

```
chicken
rabbit=(100-2*chicken)/4
```

显示如下：

```
chicken =
    22
rabbit =
    14
```

【例 4-35】利用函数的递归调用，求 $n!$ 。

```
function f=factor(n)
if n<=1
    f=1;
else
    f=factor(n-1)*n; %递归调用求(n-1)!
end
```

【例 4-36】利用函数文件，实现直角坐标 (x, y) 与极坐标 (p, θ) 之间的转换。

```
%建立函数文件 tran.m
function [rho,theta]=tran(x,y)
rho=sqrt(x*y+y*y);
theta=atan(y/x);
调用 tran.m 的脚本文件 main.m:
x=input('Please input x=:');
y=input('Please input y=:');
[rho,theta]=tran(x,y);
rho
theta
```

【例 4-37】循环实现计算变量的 \sin 、 \cos 和 \tan 值。

```
CM={'cos','sin','tan'};
for k=1:3
    theta=pi*k/12;
    y2(1,k)=eval([CM{k},'(',num2str(theta),')']);
end
```

显示如下：

```
y2 =
    0.9659    0.5000    1.0000
```

【例 4-38】nargin 用法示例。

```
%函数文件 chararray.m
function fout=chararray(a,b,c)
if nargin==1
    fout=a;
elseif nargin==2
```

```

    fout=a+b;
elseif nargin==3
    fout=(a*b*c)/2;
end
%脚本文件 mydemo.m
x=[1:3];
y=[1;2;3];
a1=chararray(x)
a2=chararray(x,y')
a3=chararray(x,y,3)

```

上例用不同的参数个数调用了 `chararray` 函数。显示如下：

```

a1 =
     1     2     3
a2 =
     2     4     6
a3 =  21

```

【例 4-39】串演算函数 `eval` 的应用。

```

clear;
t=0:0.01*pi:2*pi;
p=[1 2 4 10];
for n=1:4
    eval(['T=p(n);']) %T 依次等于数组 p 中的数值
    y=sin(T*t);
    subplot(2,2,n) %按顺序选择 4 个子图进行绘图
    plot(t,y)
    title(['y=sin(',num2str(T),'*t)'])
end

```

`eval` 函数输出结果如图 4-3 所示。

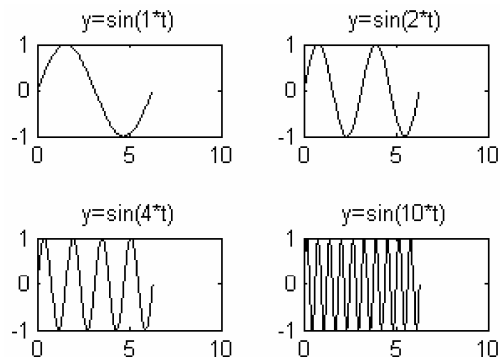


图 4-3 `eval` 函数输出结果

【例 4-40】Fibonacci 数列定义如下：

$$\begin{cases} f_1 = 1 \\ f_2 = 1 \\ f_n = f_{n-1} + f_{n-2} \quad (n > 2) \end{cases}$$

，求 Fibonacci 数列的第 20 项 f_{20} 。

代码如下：

```
function f=fibonaccii(n)
if n==0||n==1
    f=1;
else
    f=fibonaccii(n-1)+fibonaccii(n-2);
end
```

【例 4-41】绘制 $y = 1 - \frac{1}{\beta} e^{-\xi t} \sin(\beta t + \theta)$ ， ξ 分别为 0.2, 0.4, 0.6, 0.8， $t = [0, 18]$ 的曲线。

代码如下：

```
t=[0:0.1:18]';
for x=0.2:0.2:0.8
    b=sqrt([1-x^2]);
    z=atan(b/x);
    y1=t*x;
    y2=t*b+z;
    y=1-exp(y1).*sin(y2)/b;
    plot(t,y),hold on
end
xlabel('t(秒)');ylabel('y');
title('二阶系统阶跃响应');
text(3.3,0.9,'\xi=0.8')
text(4.3,1.4,'\xi=0.2')
```

二阶系统阶跃响应结果图如图 4-4 所示。

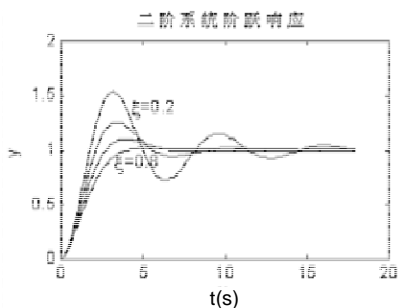


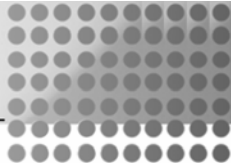
图 4-4 二阶系统阶跃响应结果图

4.3 函数类型

MATLAB 提供了多种类型的函数，有主函数、子函数、嵌套函数、私有函数和重载函数。

4.3.1 主函数

主函数在结构上与其他函数没有一点区别，之所以叫它主函数，就是因为它在 M 文件中坐了第一把交椅，其他函数都排在它后面。按惯例它与 M 文件同名，在命令窗口或者其他函数中调用这个函数，都是引用 M 文件名。也有例外，当主函数与 M 文件不同名时，只能用文件名引用这个函数。M 文件上的其他函数扮演着它的子函数角色。



4.3.2 子函数

一个 M 文件上排在主函数后面的都叫子函数，子函数的排列无规定顺序。子函数只能被同一个文件上的主函数或其他子函数调用。子函数与主函数没有形式上的区别，每个子函数都有自己的函数定义行。

例如：三个函数放在 M 文件 newexample.m 中。

```
function [avg,med]=newexample(u) %主函数
% NEWEXAMPLE Find mean and median with internal function
n=length(u);
avg=mean(u,n);
med=median(u,n);

function m=median(v,n) %子函数
% Calculate average.
a=sum(v)/n;

function m=median(v,n) %子函数
% Calculate median.
w=sort(v);
if rem(n,2)==1
    m=w((n+1)/2);
else
    m=(w(n/2)+w(n/2+1))/2;
end
```

newexample 是主函数，mean 和 median 是子函数。

几个子函数虽然在同一个文件中，但各有自己的变量，子函数之间不能相互存取别人的变量。当然，要是声明变量为全局变量，那另当别论。

1. 调用一个子函数时，寻找的顺序

不同类型的函数可以有相同的名字，而且可能放在不同的地方。因此，从 M 文件上调用一个子函数，就得从系统中寻找。MATLAB 有一个原则：首先看被调用的函数是否是本 M 文件上的子函数，是，调用它；不是，再寻找同名的私有函数。最后，从搜索路径中找内部函数。

2. 子函数的帮助文本

可以为子函数写帮助文本，与为主函数写帮助文本一样。但是，显示子函数的帮助文本与主函数的帮助文本的区别是要把 M 文件名加在子函数名前面。

假如子函数名为 mysubfun，放在 myfun.m 文件中。要在命令行得到它的帮助信息，需输入命令：

```
help myfun/mysubfun
```

4.3.3 私有函数

私有函数实际上是另一种子函数，它是私有的，只有父亲 M 文件函数和脚本能存取它。它放在一个特别名字的子目录——private 下。

① 调用私有函数的函数放在一个 M 文件上，而这个 M 文件所在的目录下直接有一个名为 private 的子目录，被调用的私有函数就在此子目录下。这就形成了父对子的调用。



② 调用私有函数的脚本，它本身要被一个 M 文件函数调用，而 M 文件函数也可以访问这个私有函数。当然，私有函数所在的 `private` 子目录，应该直接在 M 文件所在的目录下。

③ 假如私有函数名为 `myprivfun`，为了得到私有函数的帮助信息，需要输入命令：

```
help private/myprivfun
```

4.3.4 嵌套函数

嵌套函数，是指在函数里面定义函数。

1. 写嵌套函数

MATLAB 允许在 M 文件函数体中定义一个或多个函数。像任何 M 文件函数一样，被嵌套的函数能包含任何构成 M 文件的成分，正所谓麻雀虽小，五脏俱全。

只要是嵌套函数，无论是嵌套的还是被嵌套的，都以 `end` 语句结束。而且在一个 M 文件上，只要定义了嵌套函数，其他非嵌套函数也要以 `end` 语句结束。这一点可马虎不得。

嵌套是有层次的，即有不同的级别。

① 函数 A 嵌套了函数 B 和 C，B 和 C 同级。

```
function x=A(p1,p2)
...
function y=B(p3)
...
end

function z=C(p4)
...
end
...
end
```

② 多层嵌套，函数 A 嵌套 B，B 嵌套 C。A 是第一层，B 是第二层，C 是第三层。

```
function x=A(p1,p2)
...
    function y=B(p3)
        ...
            function z=C(p4)
                ...
            end
        ...
    end
...
end
```

2. 调用嵌套函数

调用嵌套函数要遵守一定的原则，原则是什么呢？请先看一个嵌套函数的示例，用它们来说明原则。

```
function A(x,y)    %主函数
B(x,y);
```

```

D(y);
function B(x,y)    % 嵌套 A
C(x);
D(y);
function C(x)      % 嵌套 B
D(x);
end
end
function D(x)      % 嵌套 A
E(x);
function E         % 嵌套 E
...
end
end
end
end

```

① 上级只能调用紧跟的下级。父亲可以调用儿子，但不能调用孙子。在上面的例子中，A 能够调用 B 和 D，不能调用 C 和 E。

② 同级能相互调用。一个父亲下的兄弟可以互相调用，即 B 和 D 可以互相调用。

③ 最下级能调用其他所有上级。孙辈可以调用他的父辈，包括他的父亲和叔叔，也能调用他的祖父，但不能调用他的堂兄弟。即 C 能调用 B、D 和 A，但不能调用 E；同样，E 也能调用 B、D 和 A，却不能调用 C。

除上述嵌套函数的调用关系外，任何嵌套函数都能调用同一个 M 文件上的子函数。

按照上述原则，给出一个实际的示例，结果如何，请读者自行动手试试。

```

function A(x,y,z)
aa=x+y+z
B(10,20);
D(30);
function B(b1,b2)
bb=b1*b2
C(12);
D(15);
function C(c1)
cc=c1/5
D(10);
end
end
function D(d1)
E(65);
function E(e1)
2*e1;
A(12,22,32)
end
end
end
end

```

3. 嵌套函数中变量的使用范围

通常，在函数之间，局部变量是不能共享的。子函数不能与主函数或其他子函数共享变量，因为每个函数都有自己的工作区，存放自己的变量。

嵌套函数也都有自己的工作区，但因为它们是嵌套关系，有些情况下可以共享变量。下面的两个嵌套函数为例说明。

<pre>function varScope1 x=5; nestfun1 function nestfun1 nestfun2 function nestfun2 x=x+1; end end end end</pre>	<pre>function varScope2 nestfun1 function nestfun1 nestfun2 function nestfun2 x=5; end end end end end</pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

① 主函数赋值的变量，能被主函数中嵌套在任何层次的函数读或重写，如 varScope1 的 x。

② 被嵌套在里层的函数赋值的变量，任何包含这个函数的函数都能读或重写。如 varScope2 的 x。

上面的两个示例，函数是一层层嵌套的，没有兄弟关系，x 被共享。x 放在最外层函数的工作区中。

下面的示例则不同，被嵌套的两个函数是兄弟。

```
function varScope3
    nestfun1
    nestfun2
        function nestfun1
            x=5;
        end
        function nestfun2
            x=x+1;
        end
    end
end
```

外层的函数 varScope3 不认识 x，因为 x 不在它的工作区中。nestfun1 定义了 x，x 在 nestfun1 的工作区中，不能被 nestfun2 共享。因此，当 nestfun2 运行，试图改变 x 时，就会出错。

4. 嵌套函数中使用函数句柄

在“调用嵌套函数”小节中，规定了调用嵌套函数的原则，也就是说，每个嵌套函数都有调用范围。在“子函数”一节中，也说明了调用子函数的范围。那么，在规定的范围之外能不能调用函数呢？答案是肯定的。

一个示例如下：

```
function h=getCubeHandle
h=@findCube; %构造函数句柄
```

```
function cube=findCube(x)      %嵌套函数
    cube=x.^3;
end
end
```

这是嵌套函数，函数 `findCube` 在内层，在正常情况下，它只能被 `getCubeHandle` 函数调用。

在 `getCubeHandle` 函数的开始，语句 `h=@findCube;`，对 `findCube` 构造了函数句柄，并且把指针保存在输出变量 `h` 中。一旦调用 `getCubeHandle`，函数 `findCube` 的指针就被返回，被保存。

在其他 M 文件中，在其他程序中，在命令行，可以使用 `findCube` 句柄调用 `findCube` 函数。

```
>> cubeIt=getCubeHandle;
>> cubeIt(9)
ans =
    729
```

函数句柄——范围之外调用函数的法宝。但有个前提，即构造句柄时，函数句柄必须在它的相关函数的范围之内。

在嵌套函数中，内层函数与外层函数可以共享某些变量。内层函数执行时，外层函数为它提供某些变量的值。

【例 4-42】嵌套函数中的变量示例。

代码如下：

```
function h=getHandle(x)
h=@nestfun;
Ve=sum(x)
    function nestfun(Vi)
        Vl=182.11;
        y=(Vi-Vl+Ve)/15
    end
end
```

被嵌套的函数 `nestfun` 有 3 个变量，即 `Ve`、`Vi` 和 `Vl`。

- ① `Ve`：与外层函数 `getHandle` 共享，它的值也由 `getHandle` 提供。
- ② `Vi`：局部变量，`nestfun` 自己赋值。
- ③ `Vl`：输入变量，调用 `nestfun` 时传递变量值。

若是执行主函数 `getHandle`，再进入 `nestfun`，函数执行不存在问题，因为 `Ve` 的值已经有了。要是从外部使用函数句柄 `h` 调用 `nestfun` 会有问题吗？`Ve` 的值从何而来？这时候，对于函数 `nestfun`，`Ve` 被认为是外部变量。

不用担心。当执行 `getHandle` 时，为 `nestfun` 函数构造了函数句柄，并保存在变量 `h` 中。`MATLAB` 不仅保存了有关函数句柄的信息，也捆绑式地保存了变量 `Ve` 的值。因此，使用函数句柄调用函数，连同变量的值一起代入了。示例如下：

```
>> clear;
x=[11 22 33 1]
x =
    11    22    33     1
>> H=getHandle(x)
```

```

Ve =
    67
H =
    @getHandle/nestfun
>> H(76.9)
y =
   -2.5473

```

一定记住：随函数句柄保存的 `Ve` 的值是永久的，只要不重新构造函数句柄，任何时候使用函数句柄调用函数，`Ve` 的值都不变。

4.3.5 重载函数

重载函数是已经存在的函数的另外版本。原来的函数是为某种特定的数据类型设计的，当要使用另外类型的数据时，要重写原函数，使它能处理新的数据类型，但它的名字与原函数名相同。调用函数的哪个版本，取决于数据类型和参数的个数。

每个重载的 MATLAB 函数，都有一个 M 文件放在 MATLAB 目录中。同一种数据类型的不同的重载函数 M 文件放在同一个目录下，目录以这种数据类型命名，并用 @ 符号开头。而不同数据类型的重载函数，放在用 MATLAB 数据类型的识别符作为名字、以 @ 符号开头的子目录中，如 `plus`。

使用 `which` 命令，选择 -all 参数，MATLAB 能够显示所指定函数的所有重载函数。

```

>> which -all plus
built-in (D:\MATLABR2008\toolbox\matlab\ops\@single\plus)      % single method
built-in (D:\MATLABR2008\toolbox\matlab\ops\@double\plus)     % double method
built-in (D:\MATLABR2008\toolbox\matlab\ops\@char\plus)       % char method
.....
D:\MATLABR2008\toolbox\wavelet\wavelet\@laurpoly\plus.m      % laurpoly method
D:\MATLABR2008\toolbox\wavelet\wavelet\@laurmat\plu          % laurmat method

```

使用者可以重载任何函数，用来处理指定的数据类型，然后把重载的函数版本放在指定数据类型目录中。

匿名函数即为无名函数，其构造极其简单，在此不再展开介绍，读者可以参考其相关资料，进一步了解。

4.4 调试程序

程序调试是程序设计的重要环节，也是程序设计人员必须掌握的重要技能。MATLAB 提供了相应的程序调试功能，既可以通过文本编辑器对程序进行调试，又可以在命令窗口结合具体的命令进行。

4.4.1 调试程序介绍

一般来说，应用程序的错误有两类：一类是语法错误，另一类是运行时的错误。语法错误包括词法或文法的错误，例如函数名拼写错误、表达式书写错误等。MATLAB 能够检查出大部分的语法错误，给出相应错误信息，并标出错误在程序中的位置。例如，输入下列程序：

```
>> a=97;
b=9.7;
c=a*b;
??? c=a*b;
```

Error: Unexpected MATLAB operator.

显然，语句 `c=a*b;` 中的 “+” 后不能接 “*” 运算符。通过分析 MATLAB 给出的错误信息，不难排查程序中的语法错误。

程序运行时的错误是指程序的运行结果有错误，这类错误也称为程序逻辑错误。MATLAB 程序运行时的错误是指程序的运行结果有错误，这类错误也称为程序逻辑错误。MATLAB 系统对逻辑错误是无能为力的，不会给出任何提示信息。这时可以通过一些调试手段来发现程序中的逻辑错误，最常见的办法是通过获取中间结果的方式来获取错误可能发生的程序段，以便进一步分析错误的原因。采取的办法如下：

① 将程序的一些主要中间结果输出到命令窗口，从而确定错误的区段。这只要去掉有关语句最后的分号即可。有时候，为了集中精力分析主要中间结果或某一程序段，可以暂时忽略一些次要语句或其他程序段，这只要在需要忽略的语句前加注释符即可。

② 使用 MATLAB 的调试菜单 (Debug)，通过图形界面操作来实现程序调试，包括设置断点、控制程序单步运行等操作。

③ 使用键盘终止函数 `keyboard` 中断程序的运行，此时将使程序的运行处于调试状态，命令窗口的提示符相应变成 `K>>`，利用命令操作方式来实现程序调试。

4.4.2 MATLAB 调试菜单

MATLAB 的 M 文件编辑器除了能编辑修改文件外，还能对程序进行调试。通过调试菜单可以查看和修改函数工作空间中的变量，从而准确地找到运行错误。通过调试菜单设置断点可以使程序运行到某一行暂停运行，这时可以查看和修改各个工作空间中的变量。通过调试菜单可以一行一行地运行程序。Debug 菜单项，如图 4-5 所示。

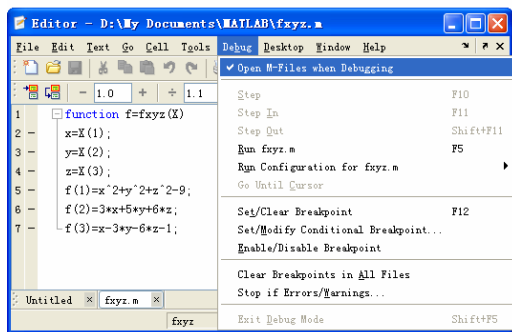


图 4-5 Debug 菜单项

1. 控制单步运行

控制单步运行的菜单命令共有 5 个。在程序运行之前，仅有 `Run file.m` 命令是激活的。只有当程序中设置了断点，且程序停止在第一个断点处时其他菜单命令才被激活，这些菜单命令如下。

- ① **Step:** 单步运行。每单击一次，程序运行一次，但不进入函数。
- ② **Step In:** 单步运行。遇到函数时进入函数内，仍单步运行。
- ③ **Step Out:** 停止单击运行。如果是在函数中，跳出函数；如果不在函数中，直接运行到下一个断点处。
- ④ **Save and Run:** 存储文件并开始运行。如果文件是已经存储过的，该选项变为 **Run**，当暂停在断点处时，该选项变为 **Continue**。
- ⑤ **Go Until Cursor:** 直接运行到光标所在的位置。

2. 断点操作

有关断点操作的菜单命令共有 5 个，包括在程序中设置和清除断点以及设置停止条件。

- ① **Set/Clear Breakpoint:** 设置或清除断点。
- ② **Set/Modify Conditional Breakpoint:** 设置或修改条件断点。条件断点可以使程序执行到满足一定条件时停止。
- ③ **Enable/Disable Breakpoint:** 使断点有效或无效。
- ④ **Clear Breakpoint in All Files:** 清除所有断点。
- ⑤ **Stop If Errors/Warnings:** 在程序执行出现错误或警告时，停止程序运行，进入调试状态，不包括 `try...catch` 语句中的错误。

4.4.3 调试命令

除了采用调试菜单调试程序外，MATLAB 还提供了一些命令用于程序调试。调试命令及其功能如表 4-1 所示。

表 4-1 调试命令及其功能

命 令	功 能
<code>dbclear</code>	清除已设置好的断点
<code>dbstop</code>	设置断点
<code>dbcont</code>	继续执行
<code>dbdown/dbup</code>	修改当前工作空间的上、下文关系
<code>dbstack</code>	显示当前堆栈的状态
<code>dbquit</code>	退出调试状态
<code>dbstatus</code>	显示所有的已设置的断点
<code>dbtype</code>	显示 M 文件代码和相应的行号
<code>dbstep</code>	执行应用程序的一行或者多行代码

表中命令的功能和调试菜单命令类似，具体使用方法请读者查询 MATLAB 帮助文档。以下一个示例说明调试命令的用法。

【例 4-43】求[1, 499]之间的全部完整数。

```
%存储程序名为 perfect.m
s=0;
for m=1:499
    for k=1:m/2
        if rem(m,k)==0
```

```

        s=s+k;
    end
end
if m==s
    disp(m);
end
end
end

```

程序运行后无任何输出，试用调试命令调试程序。

程序运行后无任何输出，说明第 2 个 if 语句中的条件 $m==s$ 不成立，由此怀疑求 s 的值不正确。为了分析方便，把 m 循环终值改为 6，即仅循环 6 次。在第 2 个 if 语句（即第 8 行）加断点，并查看 s 的值，在命令窗口输入：

```

>> dbstop in perfect at 8 %在 perfect.m 的第 8 行设置断点
>> perfect %运行程序
8      if m==s
K>> s %查看 s 的值
s =
    0
K>> dbcont %继续运行
8      if m==s
K>> s
s =
    1
.....
K>> dbcont
8      if m==s
K>> s
s =
    6
K>> dbcont
8      if m==s
K>> s
s =
   12

```

m 循环终值改为 6，共循环 6 次。第 6 次循环 ($m=6$) 时，应该求 6 的因子之和，输出结果为 12，显然错误。分析发现，第 7 次循环时， s 的初值为 6（即第 5 次循环后 s 的值）而非 0，再加上 6 的因子刚好等于 12，所以 s 赋初值的语句放错了位置，应放在 m 循环与 k 循环之间，修改如下：

```

for m=1:499
    s=0;
    for k=1:m/2
        if rem(m,k)==0
            s=s+k;
        end
    end
end

```



```
end
if m==s
    disp(m);
end
end
```

此程序运行结果前面已介绍过，请读者参阅。

第5章 MATLAB绘图功能

强大的绘图功能是 MATLAB 的特点之一。MATLAB 提供了一系列的绘图函数，用户不需要过多考虑绘图细节，只需要给出一些基本参数就能得到所需图形，这一类函数称为高层绘图函数。除此之外，MATLAB 还提供了直接对图形句柄进行操作的低层绘图操作。这类操作将图形的每个图形元素（如坐标轴、曲线、曲面或文字等）看成是一个独立的对象，系统给每个图形对象分配一个句柄，以后可以通过该句柄对该图形元素进行操作，而不影响图形的其他部分。高层绘图操作简单明了、方便高效，是用户最常使用的绘图方法，而低层绘图操作控制和表现图形的能力更强，为用户更加自主地绘制图形创造了条件。事实上，MATLAB 的高层绘图函数都是利用低层绘图函数而建立起来的。

5.1 二维图形绘制

二维图形是将平面坐标上的数据点连接起来的平面图形。可以采用不同的坐标系，除直角坐标系外，还可采用对数坐标、极坐标。数据点可以用向量或矩阵形式给出，类型可以是实型或复型。二维图形的绘制无疑是其他绘图操作的基础。

5.1.1 绘制二维曲线的常用函数

在 MATLAB 中，最基本且应用最为广泛的绘图函数为 plot 函数，利用它可以在二维平面上绘制出不同的曲线。

1. plot函数的基本用法

plot 函数用于绘制 xy 平面上的线性坐标曲线图，因此需要提供一组 x 坐标及其各点对应的 y 坐标，这样就可以绘制分别以 x 和 y 为横、纵坐标的二维曲线。plot 函数的基本调用格式为：

```
plot(x,y)
```

其中， x 和 y 为长度相同的向量，分别用于存储 x 坐标和 y 坐标数据。

【例 5-1】在 $0 \leq x \leq 2\pi$ 区间内，绘制曲线 $y = 2e^{-0.5x} \sin(2\pi x)$ 。

代码如下：

```
x=0:pi/100:2*pi;  
y=2*exp(-0.5*x).*sin(2*pi*x);  
plot(x,y)
```

程序执行后，打开一个图形窗口，在其中绘制如图 5-1 所示的曲线。

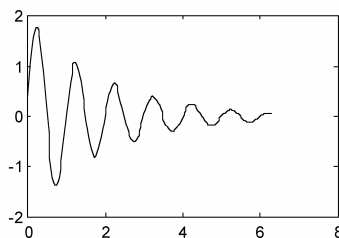


图 5-1 $y = 2e^{-0.5x} \sin(2\pi x)$ 的曲线

【例 5-2】绘制曲线

$$\begin{cases} x = t \cos(3t) \\ y = t \sin^2 t \end{cases} \quad -\pi \leq t \leq \pi$$

这是以参数方程形式给出的二维曲线，只要给定参数向量，再分别求出 x ， y 即可绘出曲线。代码如下：

```
t=-pi:pi/100:pi;
x=t.*cos(3*t);
y=t.*sin(t).*(sin(t));
plot(x,y);
```

运行程序，得到图形如图 5-2 所示。

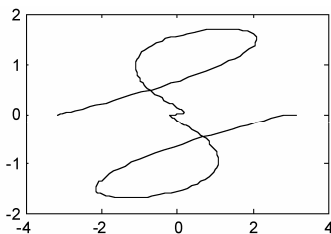


图 5-2 以参数方程形式给出的二维曲线

以上提到 `plot` 函数的自变量 x 、 y 为长度相同的向量，这是最常见和最基本的情况。实际应用中还有一些变化，下面分别说明。

① 当 x 是向量， y 是与 x 同维的矩阵时，则绘制出多根不同色彩的曲线。线条数等于 y 矩阵的另一维数， x 被作为这些曲线共同的横坐标。例如，下列程序可以在同一坐标中同时绘制出正弦和余弦曲线。

```
x=linspace(0,2*pi,100);
y=[sin(x);cos(x)];
plot(x,y)
```

程序首先产生一个行向量 x ，然后分别求取行向量 $\sin(x)$ 和 $\cos(x)$ ，并将它们构成矩阵 y 的两行，最后在同一坐标中同时绘制两条曲线。

② 当 x 、 y 是同维矩阵时，则以 x 、 y 对应列元素为横、纵坐标分别绘制曲线，线条数等于矩阵的列数。例如，在同一坐标中同时绘制出正弦和余弦曲线，可用下面的程序。

```
t=linspace(0,2*pi,100);
x=[t;t]';
y=[sin(t);cos(t)];
plot(x,y)
```

③ `plot` 函数最简单的调用格式是只包含一个输入参数。

```
plot(x)
```

在这种情况下，当 x 是实向量时，则以该向量元素的下标为横坐标，元素值为纵坐标画出一条曲线，这实际上是绘制折线图。当 x 是复数向量时，则分别以该向量元素实部和虚部为横、纵坐标绘制出一条曲线。例如，下面的程序可以绘制一个单位圆。

```
t=0:0.01:2*pi;
x=exp(i*t); %x 是一个复数向量
plot(x)
```



程序中的 i 是虚数单位, 这样 x 是一个复数向量。为了保证这一点, i 不能被赋予其他的值。

当 x 是实矩阵时, 则按列绘制每列元素值相对其下标的曲线, 曲线数等于 x 阵的列数。当输入参数是复数矩阵时, 则按列分别以元素实部和虚部为横、纵坐标绘制多条曲线。例如, 下面的程序可以绘制 3 个同心圆。

```
t=0:0.01:2*pi;
x=exp(i*t);
y=[x;2*x;3*x]';
plot(y)
```

2. 含多个输入参数的plot函数

plot 函数可以包含若干组向量对, 每一向量对可以绘制出一条曲线。含多个输入参数的 plot 函数调用格式为:

```
plot(x1, y1, x2, y2, ..., xn, yn)
```

① 当输入参数都为向量时, $x1$ 和 $y1$, $x2$ 和 $y2$, ..., xn 和 yn 分别组成一组向量对, 每一组向量对的长度可以不同。每一向量对可以绘制出一条曲线, 这样可以在同一坐标内绘制出多条曲线。例如, 下面程序可以在同一坐标中同时绘制出 3 根正弦曲线。

```
x=linspace(0,2*pi,100);
plot(x,sin(x),x,2*sin(x),x,3*sin(x))
```

② 当输入参数有矩阵形式时, 配对的 x 、 y 按对应列元素为横、纵坐标分别绘制曲线, 曲线数等于矩阵的列数。分析下列程序绘制的曲线。

```
x=linspace(0,2*pi,100);
y1=sin(x);
y2=2*sin(x);
y3=3*sin(x);
x=[x;x;x]';
y=[y1;y2;y3]';
plot(x,y,x,cos(x))
```

x 和 y 都是含有 3 列的矩阵, 它们组成输入参数对, 绘制出 3 根正弦曲线; x 和 $\cos(x)$ 都是向量, 它们组成输入参数对, 绘制出 1 根余弦曲线。

3. 含选项的plot函数

MATLAB 提供了一些绘图选项, 用于确定所绘曲线的线型、颜色和 data 点标记符号。这些选项如表 5-1 所示, 它们可以组合使用。例如, “b-” 表示蓝色点画线, “y:d” 表示黄色虚线并用菱形符号标记数据点。当选项省略时, MATLAB 规定, 线型一律用实线, 颜色将根据曲线的先后顺序依次采用表 5-1 给出的前 7 种颜色。含选项的 plot 函数调用格式为:

```
plot(x1,y1,选项 1, x2, y2, 选项 2, ..., xn, yn, 选项 n)
```

表 5-1 线型、颜色和标记符号选项

线 型			
-	实线		
:	虚线		
-.	点画线		
--	双画线		
颜 色			
b	蓝色	m	品红色
g	绿色	y	黄色
r	红色	k	黑色
c	青色	w	白色

标识符号			
.	点	^	朝上三角符号
o	圆圈	*	星号
x	叉号	<	朝左三角符号
+	加号	>	朝右三角符号
s	方块符(square)	p	六角星符(pentagram)
d	菱形符(diamond)	h	六角星符(Hexgram)
v	朝下三角符号		

【例 5-3】用不同线型和颜色在同一坐标内绘制曲线 $y = 2e^{-0.5x} \sin(2\pi x)$ 及其包络线。

代码如下：

```
clear;
x=(0:pi/100:2*pi)';
y1=2*exp(-0.5*x)*[1,-1];
y2=2*exp(-0.5*x).*sin(2*pi*x);
x1=(0:12)/2;
y3=2*exp(-0.5*x1).*sin(2*pi*x1);
plot(x,y1,'m-',x,y2,'b--',x1,y3,'rp');
```

运行程序，效果如图 5-3 所示。

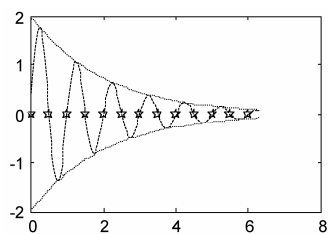


图 5-3 用不同线型和颜色绘制曲线效果

4. 双纵坐标函数plotyy

在 MATLAB 中，如果需要绘制出具有不同纵坐标度的两个图形，可以使用 plotyy 函数。这种图形能把函数值具有不同量纲、不同数量级的两个函数绘制在同一坐标中，有利于图形数据的对比分析。plotyy 函数的调用格式为：

plotyy(x1, y1, x2, y2)

其中，x1, y1 对应一条曲线，x2, y2 对应另一条曲线。横坐标的标度相同，纵坐标有两个，左纵坐标用于 x1, y1 数据对，右纵坐标用于 x2, y2 数据对。

【例 5-4】用不同标度在同一坐标内绘制曲线 $y_1 = e^{-0.5x} \sin(2\pi x)$ 及曲线 $y_2 = 1.5e^{-0.2x} \sin x$ 。

代码如下：

```
clear;
x1=0:pi/100:2*pi;
x2=0:pi/100:3*pi;
y1=exp(-0.5*x1).*sin(2*pi*x1);
y2=1.5*exp(-0.2*x2).*sin(x2);
plotyy(x1,y1,x2,y2);
```

运行程序，效果如图 5-4 所示。

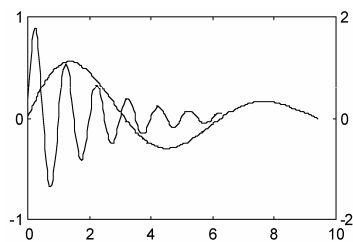


图 5-4 用双纵坐标绘制的曲线

5.1.2 绘制图形的辅助操作

绘制完图形后，可能还需要对图形进行一些辅助操作，以使图形意义更加明确，可读性更强。

1. 图形标注

在绘制图形的同时，可以对图形加上一些说明，如图形名称、坐标轴说明以及图形某一部分的含义等，这些操作称为添加图形标注。有关图形标注函数的调用格式为：

```
xlabel (x 轴说明)
ylabel (y 轴说明)
title (图形名称)
text (x, y, 图形说明)
legend (图例 1, 图例 2, ...)
```

其中，xlabel、ylabel 和 title 函数分别用于说明坐标轴的名称和图形。text 函数是在 (x, y) 坐标处添加图形说明。添加文本说明也可用 gtext 命令，执行该命令时，十字坐标光标自动跟随鼠标移动，单击鼠标即可将文本放置在十字光标处，如命令 gtext('cos(x)'), 即可放置字符串 cos(x)。legend 函数用于绘制曲线所用线型、颜色或数据点标记图例，图例放置在图形空白处，用户还可以通过鼠标移动图例，将其放到所希望的位置。除 legend 函数外，其他函数同样适用于三维图形，z 坐标轴说明用 zlabel 函数。

2. 坐标控制

在绘制图形时，MATLAB 可以自动根据要绘制曲线数据的范围选择合适的坐标刻度，使得曲线能够尽可能清晰地显示出来。所以，在一般情况下用户不必选择坐标轴的刻度范围。但是，如果用户对坐标系不满意，可利用 axis 函数对其重新设定。该函数的调用格式为：

```
axis([xmin xmax ymin ymax zmin zmax])
```

如果只给出前 4 个参数，则 MATLAB 按照给出的 x、y 轴的最小值和最大值选择坐标系范围，以便绘制出合适的二维曲线。如果给出了全部参数，则系统按照给出的 3 个坐标轴的最小值和最大值选择坐标系范围，以便绘制出合适的三维图形。

axis 函数功能丰富，常用的用法如下。

axis equal: 纵、横坐标轴采用等长刻度。

axis square: 产生正方形坐标系（默认为矩形）。

axis auto: 使用默认设置。

axis off: 取消坐标轴。

axis on: 显示坐标轴。

给坐标加网格线用 grid 命令来控制。grid on/off 命令控制是否画网格线，不带参数的 grid 命令在两种状态之间进行切换。

给坐标加边框用 box 命令来控制。box on/off 命令控制是否加边框线，不带参数的 box 命令在两种状态之间进行切换。

【例 5-5】绘制分段函数曲线并添加图形标注。

$$f(x) = \begin{cases} \sqrt{x}, & 0 \leq x < 4 \\ 2, & 4 \leq x < 6 \\ 5 - x/2, & 6 \leq x < 8 \\ 1, & x \geq 8 \end{cases}$$

代码如下：

```

clear;
x1=0:pi/100:2*pi;
x2=0:pi/100:3*pi;
y1=exp(-0.5*x1).*sin(2*pi*x1);
y2=1.5*exp(-0.2*x2).*sin(x2);
plotyy(x1,y1,x2,y2);
>> clear;
x=linspace(0,10,100);
y=[];
for x1=x
    if x1>=8
        y=[y,1];
    elseif x1>=6
        y=[y,5-x1/2];
    elseif x1>=4
        y=[y,2];
    elseif x1>=0
        y=[y,sqrt(x1)];
    end
end
plot(x,y)
axis([0 10 0 2.5]);      %设置坐标轴
title('分段函数曲线');   %加图形标题
xlabel('Variable x');     %加 x 轴说明
ylabel('Variable y');     %加 y 轴说明
text(2,1.3,'y=x^{1/2}'); %在指定位置添加图形说明
text(4.5,1.9,'y=2');
text(7.3,1.5,'y=5-x/2');
text(8.5,0.9,'y=1');

```

运行程序，效果如图 5-5 所示。

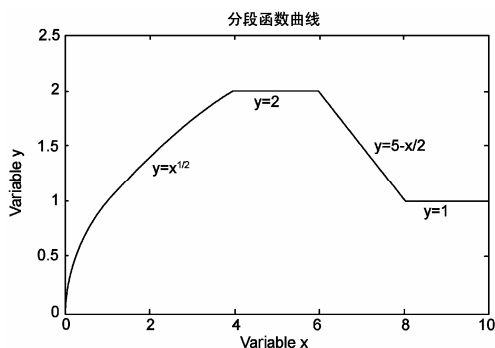


图 5-5 给图形添加图形标注

3. 图形保持

一般情况下，每执行一次绘图命令，就刷新一次当前图形窗口，图形窗口原有图形将不复存在。若希望在已存在的图形上再继续添加新的图形，可使用图形保持命令 **hold**。**hold on/off** 命令控制是保持原有图形还是刷新原有图形，不带参数的 **hold** 命令在两种状态之间进行切换。

【例 5-6】用图形保持功能在同一坐标内绘制曲线 $y = 1.5e^{-0.5x} \sin(2\pi x)$ 及其包络线。
代码如下：

```
clear;
x=(0:pi/100:2*pi)';
y1=1.5*exp(-0.5*x)*[1,-1];
y2=1.5*exp(-0.5*x).*sin(2*pi*x);
plot(x,y1,'b+');
axis([0,1.5*pi,-1.5,1.5]); % 设置坐标
hold on; % 设置图形保持状态
plot(x,y2,'r');
legend('包络线','包络线','曲线 y'); % 加图例
hold off; % 关闭图形保持
grid on
```

运行程序，效果如图 5-6 所示。

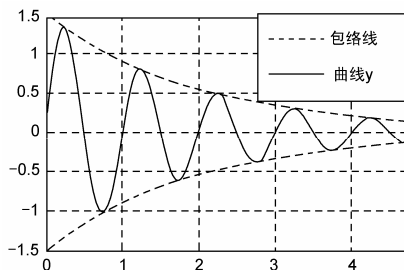


图 5-6 利用图形保持绘制多根曲线

4. 图形窗口的分割

在实际应用中，经常需要在一个图形窗口内绘制若干个独立的图形，这就需要对图形窗口进行分割。分割后的图形窗口由若干个绘图区组成，每一个绘图区可以建立独立的坐标系并绘制图形。同一图形窗口中的不同图形称为子图。MATLAB 系统提供了 subplot 函数，用来将当前图形窗口分割成若干个绘图区。每个区域代表一个独立的子图，也是一个独立的坐标系，可以通过 subplot 函数激活某一区，该区为活动区，所发出的绘图命令都作用于活动区域。subplot 函数的调用格式为：

```
subplot(m,n,p)
```

该函数将当前图形窗口分成 $m \times n$ 个绘图区，即 m 行，每行 n 个绘图区，区号按行优先编号，且选定第 p 个区为当前活动区。在每一个绘图区允许以不同的坐标系单独绘制图形。

【例 5-7】在一个图形窗口中以子图形式同时绘制正弦、余弦、正切、余切曲线。

代码如下：

```
clear;
x=linspace(0,2*pi,60);
y=sin(x);
z=cos(x);
t1=sin(x)./(cos(x)+eps);
t2=cos(x)./(sin(x)+eps);
subplot(2,2,1);
```



```

plot(x,y);
title('sin(x)');
axis([0,2*pi,-1,1]);
subplot(2,2,2);
plot(x,z);
title('cos(x)');
axis([0,2*pi,-1,1]);
subplot(2,2,3);
plot(x,t1);
title('tangent(x)');
axis([0,2*pi,-40,40]);
subplot(2,2,4);
plot(x,t2);
title('cotangent(x)');
axis([0,2*pi,-1,1]);

```

运行程序，效果如图 5-7 所示。

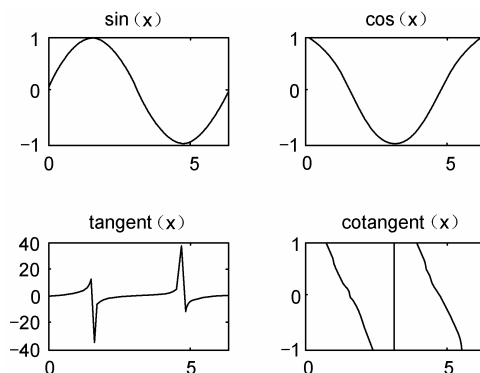


图 5-7 图形窗口的分割

例中将图形窗口分割成 2×2 个绘图区，编号从 1 到 4，各区分别绘制一幅图形，这是最规则的情况。实际上，还可以做更灵活的分割。示例如下：

```

clear;
x=linspace(0,2*pi,60);
y=sin(x);
z=cos(x);
t1=sin(x)./(cos(x)+eps);
t2=cos(x)./(sin(x)+eps);
subplot(2,2,1);           %选择 2×2 区中的 1 号区
stairs(x,y);
title('sin(x)-1');
axis([0,2*pi,-1,1]);
subplot(2,1,2);           %选择 2×1 区中的 2 号区
stem(x,y);
title('sin(x)-2');
axis([0,2*pi,-1,1]);
subplot(4,4,3);           %选择 4×4 区中的 3 号区
plot(x,y);

```

```

title('sin(x)');
axis([0,2*pi,-1,1]);
subplot(4,4,4);           %选择 4×4 区中的 4 号区
plot(x,z);
title('cos(x)');
axis([0,2*pi,-1,1]);
subplot(4,4,7);           %选择 4×4 区中的 7 号区
plot(x,t1);
title('tangent(x)');
axis([0,2*pi,-40,40]);
subplot(4,4,8);           %选择 4×4 区中的 8 号区
plot(x,t2);
title('cotangent(x)');
axis([0,2*pi,-40,40]);

```

程序运行结果如图 5-8 所示。利用坐标轴对象操作可以对图形窗口进行任何分割。

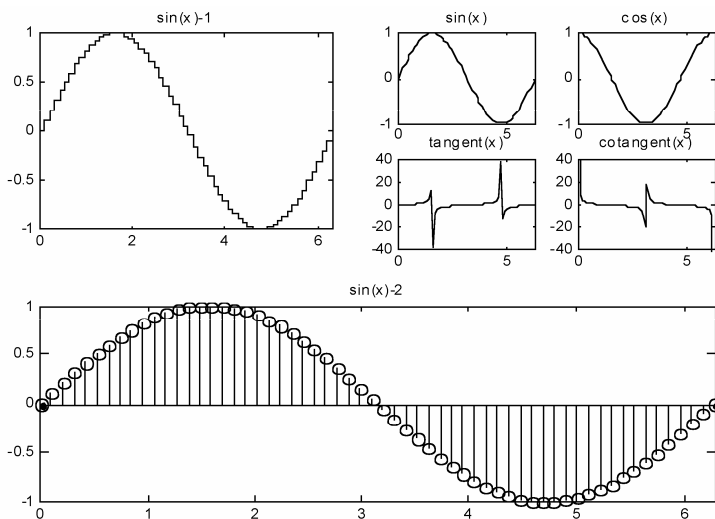


图 5-8 图形窗口的灵活分割

5.1.3 绘制二维图形的其他函数

1. 其他形式的线性直角坐标图

在线性直角坐标系中，其他形式的图形有条形图、阶梯图、杆图和填充图等，所采用的函数分别是：

```

bar(x, y, 选项)
stairs(x, y, 选项)
stem(x, y, 选项)
fill(x1, y1, 选项 1, x2, y2, 选项 2, ...)

```

前 3 个函数的用法与 plot 函数相似，只是没有多输入变量形式。fill 函数按向量元素下标递增次序依次用直线段连接 x、y 对应元素定义的数据点。假如这样连接所得到的折线不封闭，那么 MATLAB 将自动把该折线的首尾连接起来，构成封闭多边形。然后将多边形内部涂满指定的颜色。

【例 5-8】分别以条形图、填充图、阶梯图和杆图形式绘制曲线 $y = 1.5e^{-0.5x} \sin x$ 。

代码如下：

```
clear;
x=0:0.35:7;
y=1.5*exp(-0.5*x);
subplot(2,2,1);
bar(x,y,'r');
title('bar(x,y,"r")');
axis([0 7 0 2]);
subplot(2,2,2);
fill(x,y,'g');
title('fill(x,y,"g")');
axis([0 7 0 2]);
subplot(2,2,3);
stairs(x,y,'b');
title('stairs(x,y,"b")');
axis([0 7 0 2]);
subplot(2,2,4);
stem(x,y,'m');
title('stem(x,y,"m")');
axis([0 7 0 2]);
```

运行程序，效果如图 5-9 所示。

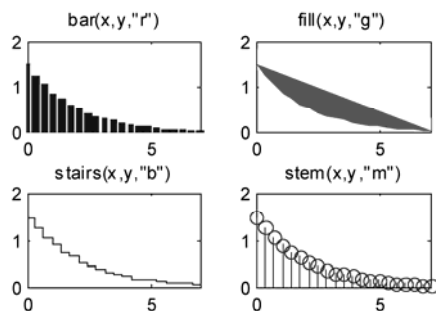


图 5-9 几种不同形式的二维图形

2. 极坐标图

polar 函数用来绘制极坐标图，其调用格式为：

```
polar(the,rho,选项)
```

其中，the 为极坐标极角，rho 为极坐标矢径，选项的内容与 plot 函数相似。

【例 5-9】绘制 $\rho = \sin(2\theta)\cos(2\theta)$ 的极坐标图。

代码如下：

```
clear;
the=0:0.01:2*pi;
rho=sin(2*the).*cos(2*the);
polar(the,rho,'r');
```

运行程序，效果如图 5-10 所示。

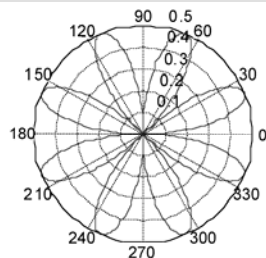


图 5-10 极坐标图

3. 对数坐标图形

在实际应用中,经常用到对数坐标,例如控制理论中的 Bode 图。MATLAB 提供了绘制对数和半对数坐标曲线的函数,调用格式为:

```
semilogx(x1,y1,选项 1,x2,y2,选项 2,...)
semilogy(x1,y1,选项 1,x2,y2,选项 2,...)
loglog(x1,y1,选项 1,x2,y2,选项 2,...)
```

其中,选项的定义与 plot 函数完全一致,所不同的是坐标轴的选取。semilogx 函数使用半对数坐标, x 轴为常用对数刻度,而 y 轴仍保持线性刻度。semilogy 函数也使用半对数坐标, y 轴为常用对数刻度,而 x 轴仍保持线性刻度。loglog 函数使用全对数坐标, x、y 轴均采用常用对数刻度。

【例 5-10】绘制 $y = 10x^2$ 的对数坐标图,并与直角线性坐标图进行比较。

代码如下:

```
clear;
x=0:0.1:10;
y=10*x.*x;
subplot(2,2,1);
plot(x,y);
title('plot(x,y)');grid on;
subplot(2,2,2);
semilogx(x,y);
title('semilogx(x,y)');grid on;
subplot(2,2,3);
semilogy(x,y);
title('semilogy(x,y)');grid on;
subplot(2,2,4);
loglog(x,y);
title('loglog(x,y)');grid on;
```

运行程序,效果如图 5-11 所示。

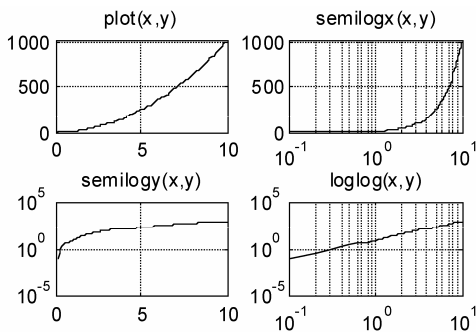


图 5-11 对数坐标图

在前面介绍过利用冒号表达式或 linspace 函数产生线性坐标向量, MATLAB 还提供了一个实用的函数 logspace, 它可以按对数等距分布来产生一个向量。该函数的调用格式为:

```
logspace(a, b, n)
```

其中, a 和 b 是生成向量的第一个和最后一个元素, n 是元素总数, 当 n 省略时, 自动产生 50 个元素。

4. 对函数自适应采样的绘图函数

前面介绍了很多绘图函数，基本的操作方法为：先取足够稠密的自变量向量，然后计算出函数值向量，最后用绘图函数绘图。在取数据点时一般都是等间隔采样，这对绘制高频率变化函数不够精确。例如函数 $f(x) = \cos(\tan(\pi x))$ ，在 $(0,1)$ 范围有无限多个振荡周期，函数变化率大。为提高精度，绘制出比较真实的函数曲线，就不能等间隔采样，而必须在变化率大的区段密集采样，以充分反映函数的实际变化规律，进而提高图形的真实度。fplot 函数可自适应地对函数进行采样，能更好地反映函数的变化规律。该函数的调用格式为：

fplot(filename, lims, tol, 选项)

其中，filename 为函数名，以字符串形式出现。它可以是由多个分量函数构成的行向量，分量函数可以是函数的直接字符串，也可以是内部函数名或函数文件名，但自变量都必须为 x。lims 为 x、y 的取值范围，以行向量形式出现，取二元向量 [xmin, xmax] 时，x 轴的范围被人为确定，取四元向量 [xmin, xmax, ymin, ymax] 时，x、y 轴的范围被人为确定。tol 为相对允许误差，其系统默认值为 $2e-3$ 。选项定义与 plot 函数相同。例如：

```
fplot('sin(x)', [0,2*pi], '*')
fplot(['sin(x),cos(x)'], [0,2*pi,-1.5,5], 'le-3, 'r') %绘制正、余弦曲线
```

观察上述语句绘制的正、余弦曲线采样点的分布，可发现曲线变化率大的区段，采样点比较密集。

【例 5-11】用 fplot 函数绘制 $f(x) = \cos(\tan(\pi x))$ 的曲线。

先建立 myf.m 文件：

```
function y=myexample(x)
y=cos(tan(pi*x));
```

再用 fplot 函数绘制 myexample.m 函数的曲线：

```
>> fplot('myexample', [-0.4,1.4], 1e-4)
```

得到如图 5-12 所示曲线。从图 5-12 中可以看出，在 $x=0.5$ 附近采样点十分密集。

也可以直接用 fplot 函数绘制 $f(x) = \cos(\tan(\pi x))$ 的曲线：

```
>> fplot('cos(tan(pi*x))', [-0.4,1.4], 1e-4)
```

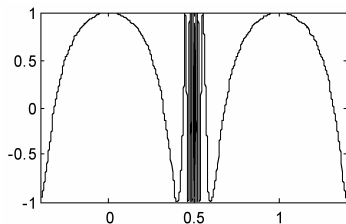


图 5-12 自适应采样绘图

5. 其他形式的二维图形

MATLAB 提供的绘图函数还有很多，例如，用来表示各元素占总和的百分比的饼图、复数的相量图等。示例如下：

① 某次考试优秀、良好、中等、不及格的人数分别为 9、21、19、18、7，试用饼图进行成绩统计分析。

② 绘制复数的相量图， $2+3i$ 、 $6.3-i$ 和 $-1.2+4i$ 。

代码如下：

```
subplot(1,2,1);
pie([9,21,19,18,7]);
title('饼图');
legend('优秀','良好','中等','及格','不及格');
subplot(1,2,2);
```

```
compass([2+3i,6.3-i,-1.2+4i]);
title('相量图');
```

运行程序，效果如图 5-13 所示。

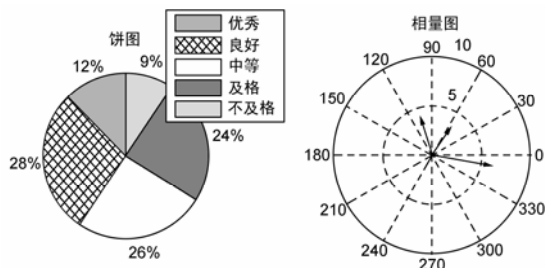


图 5-13 其他形式二维图形示例

5.2 三维图形绘制

5.2.1 绘制三维曲线的常用函数

最常用的三维图形函数为 `plot3`，它将二维绘图函数 `plot` 的有关功能扩展到三维空间，用来绘制三维曲线。`plot3` 函数与 `plot` 函数用法十分相似，其调用格式为：

```
plot3(x1,y1,z1,选项 1,x2,y2,z2,选项 2,...,xn,yn,zn,选项 n)
```

其中，每一组 x 、 y 、 z 组成一组曲线的坐标参数，选项的定义和 `plot` 函数相同。当 x 、 y 、 z 是同维向量时，则 x 、 y 、 z 对应元素构成一条三维曲线。当 x 、 y 、 z 是同维矩阵时，则以 x 、 y 、 z 对应列元素绘制三维曲线，曲线条数等于矩阵列数。

【例 5-12】绘制空间曲线

$$\begin{cases} x^2 + y^2 + z^2 = 64 \\ y + z = 0 \end{cases}$$

曲线所对应的参数方程为

$$\begin{cases} x = 10 \cos t \\ y = 3\sqrt{3} \sin t \\ z = -3\sqrt{3} \sin t \end{cases} \quad 0 \leq t \leq 2\pi$$

代码如下：

```
clear;
t=0:pi/50:2*pi;
x=8*cos(t);
y=4*sqrt(2)*sin(t);
z=-4*sqrt(2)*sin(t);
plot3(x,y,z,'o');
title('Line in 3-D Space');
text(0,0,0,'origin');
xlabel('x');ylabel('y');
zlabel('z');grid;
```

运行程序，效果如图 5-14 所示。

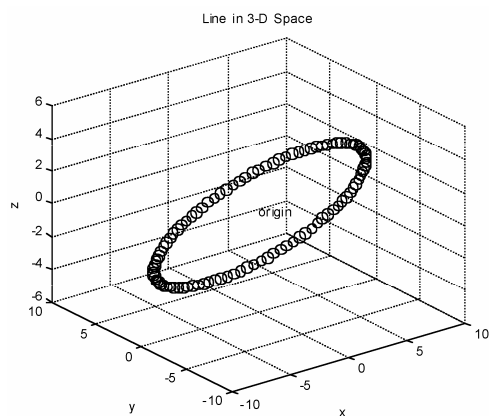


图 5-14 三维函数绘制珍珠项链

5.2.2 三维曲面图绘制

1. 平面网格坐标矩阵的生成

绘制 $z = f(x, y)$ 所代表的三维曲面图，先要在 xy 平面选定一矩形区域，假定矩形区域 $D = [a, b] \times [c, d]$ ，然后将 $[a, b]$ 在 x 方向分成 m 份，将 $[c, d]$ 在 y 方向分成 n 份，由各划分分别做平行于两坐标轴的直线，将区域 D 分成 $m \times n$ 个小矩形，生成代表每一个小矩形顶点坐标的平面网格坐标矩阵，最后利用有关函数绘图。

产生平面区域内的网格坐标矩阵有两种方法。

(1) 利用矩阵运算生成

```
x=a:dx:b;
y=c:dy:d;
X1=ones(size(y))*x;
Y1=y*ones(size(x));
```

上述语句执行后，矩阵 X 的每一行都是向量 x ，行数等于向量 y 的元素的个数，矩阵 Y 的每一列都是向量 y ，列数等于向量 x 的元素的个数。于是 X 和 Y 相同位置上的元素 $(X(i, j))$ ， $(Y(i, j))$ 恰好是区域 D 的 (i, j) 的网格点的坐标。若根据每一个网格点上的 x 、 y 坐标求函数值 z ，则得到函数值矩阵 Z 。显然， X 、 Y 、 Z 各列或各行所对应坐标，对应于一条空间曲线，空间曲线的集合组成空间曲面。

(2) 利用 meshgrid 函数生成

```
x=a:dx:b;
y=c:dy:d;
[X,Y]=meshgrid(x,y);
```

语句执行后，所得到的网格坐标矩阵 X 、 Y ，与方法 (1) 得到的相同。当 $x = y$ 时，meshgrid 函数可写成 meshgrid(x)。

【例 5-13】已知 $8 < x < 30$ ， $12 < y < 35$ ，求不定方程 $3x + 4y = 135$ 的整数解。

代码如下：

```
x=9:29;
y=13:34;
```

```
[X,Y]=meshgrid(x,y);    %在[8,29]×[13,34]区域生成网格坐标
z=3*X+4*Y;
k=find(z==135);          %找出解的位置
X(k)',Y(k)'              %输出对应位置的 X, Y, 即方程的解
```

显示如下:

```
ans =
     9    13    17    21    25
ans =
    27    24    21    18    15
```

即方程共有 5 组解: (9,27)、(13,24)、(17,21)、(21,18)、(25,15)。

2. 绘制三维曲面的函数

MATLAB 提供了 mesh 函数和 surf 函数来绘制三维曲面图。mesh 函数用于绘制三维网格图。在不需要绘制特别精细的三维曲面图时, 可以通过三维网格图来表示三维曲面。surf 用于绘制三维曲面图, 各线条之间的补面用颜色填充。mesh 函数和 surf 函数的调用格式为:

```
mesh(x, y, z, c)
surf(x, y, z, c)
```

一般情况下, x 、 y 、 z 是维数相同的矩阵。 x 、 y 是网格坐标矩阵, z 是网格点上的高度矩阵, c 用于指定在不同高度下的颜色范围。 c 省略时, MATLAB 认为 $c=z$, 亦即颜色的设定是正比图形的高度的, 这样就可以得出层次分明的三维图形。当 x 、 y 省略时, 把 z 矩阵的列下标当成 x 轴坐标, 把 z 矩阵的行下标当成 y 轴坐标, 然后绘制三维曲面图。当 x 、 y 是向量时, 要求 x 的长度必须等于 z 矩阵的列数, y 的长度等于 z 矩阵的行数, x 、 y 向量元素的组合构成网格点的 x 、 y 坐标, z 坐标则取自 z 矩阵, 然后绘制三维曲面图。

【例 5-14】用三维曲面图绘制出 $z = \sin y \cos x$ 。

为便于分析各种三维曲面的特征, 下面画出了 3 种不同形式的曲面。

代码 1:

```
clear;
x=0:0.1:2*pi;
[x,y]=meshgrid(x);
z=sin(y).*cos(x);
plot3(x,y,z);          %绘制图 5-15
xlabel('x-axis');ylabel('y-axis');zlabel('z-axis');
title('plot3-1');grid on;
```

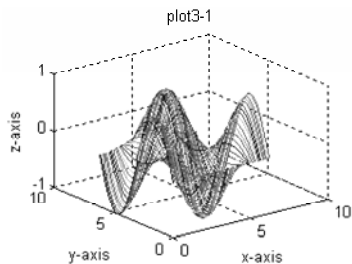


图 5-15 用 plot3 绘制的曲面图

代码 2:

```
clear;
x=0:0.1:2*pi;
[x,y]=meshgrid(x);
z=sin(y).*cos(x);
mesh(x,y,z);           %绘制图 5-16
xlabel('x-axis');ylabel('y-axis');zlabel('z-axis');
title('mesh');grid on;
```


代码 3:

```
clear;
x=0:0.1:2*pi;
[x,y]=meshgrid(x);
z=sin(y).*cos(x);
surf(x,y,z);           %绘制图 5-17
xlabel('x-axis');ylabel('y-axis');zlabel('z-axis');
title('surf');grid on;
```

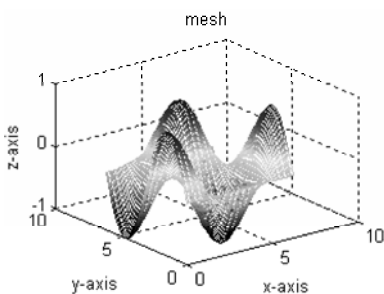


图 5-16 三维网格图

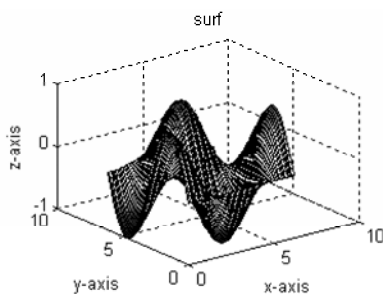


图 5-17 三维曲面图

程序执行结果分别如图 5-15、图 5-16 及图 5-17 所示。从图中可以发现，网格图（mesh）中线条有颜色，线条间的补面无颜色。曲面图（surf）的线条是黑色的，线条间的补面有颜色。还可进一步观察到曲面图的补面颜色和网格图的线条颜色都是沿 z 轴变化的。用 `plot3` 绘制的三维曲面实际上由三维曲线组合而成。

【例 5-15】绘制两个直径相等的圆管的相交图形。

代码如下:

```
%两个等直径圆管的交线
m=35;
z=1.5*(0:m)/m;
r=ones(size(z));
the=(0:m)/m*2*pi;
x1=r*cos(the);
y1=r*sin(the); %生成第一个圆管的坐标矩阵
z1=z*ones(1,m+1);
x=(-m:2:m)/m;
x2=x*ones(1,m+1);
y2=r*cos(the); %生成第二个圆管的坐标矩阵
z2=r*sin(the);
surf(x1,y1,z1); %绘制竖立的圆管
axis equal;axis off;
hold on;
surf(x2,y2,z2);
axis equal;axis off;
title('两个圆管的交线');
hold off;
```

两个圆管的交线

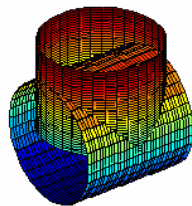


图 5-18 两个圆管的交线

运行程序，效果如图 5-18 所示。

函数 surf 也有两个类似的函数,即具有等高线的曲面函数 surfc 和具有光照效果的曲面函数 surfl。

【例 5-16】在 xy 平面内选择区域 $[-7,7] \times [-7,7]$, 绘制函数

$$z = \frac{\sin \sqrt{x^2 + y^2}}{\sqrt{x^2 + y^2}}$$

的 4 种三维曲面图。

代码如下:

```
[x,y]=meshgrid(-7:0.5:7);
z=sin(sqrt(x.^2+y.^2))./sqrt(x.^2+y.^2+eps);
subplot(2,2,1);
surfc(x,y,z);
title('surfc(x,y,z)');
subplot(2,2,2);
surfl(x,y,z);
title('surfl(x,y,z)');
subplot(2,2,3);
meshc(x,y,z);
title('meshc(x,y,z)');
subplot(2,2,4);
meshz(x,y,z);
title('meshz(x,y,z)');
```

运行程序, 效果如图 5-19 所示。

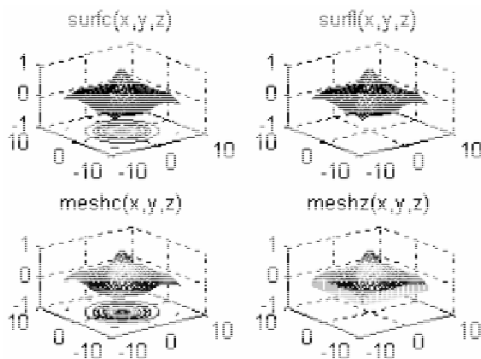


图 5-19 4 种形式的三维曲面图

3. 标准三维曲面

MATLAB 提供了一些函数用于绘制标准三维曲面, 这些函数可以产生相应的绘图数据, 常用于三维图形的演示。例如, sphere 函数和 cylinder 函数分别适用于绘制三维球面和柱面。其调用格式参看以下示例。

MATLAB 还有一个 peaks 函数, 称为多峰函数, 常用于三维曲面的演示。该函数可以用来生成绘制数据矩阵, 矩阵元素由函数

$$f(x, y) = 3(1 - x^2)e^{-x^2 - (y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2 - y^2} - \frac{1}{3}e^{-(x+1)^2 - y^2}$$

在矩形区域 $[-3,3] \times [-3,3]$ 的等分网格上的函数值确定。调用格式参看以下示例。

【例 5-17】绘制标准三维曲面图形。

代码如下：

```
t=0:pi/15:2*pi;
[x,y,z]=sphere;
subplot(1,3,1);
surf(x,y,z);
subplot(1,3,2);
[x,y,z]=cylinder(2+sin(t),30);
surf(x,y,z);
subplot(1,3,3);
[x,y,z]=peaks(30);
meshz(x,y,z);
```

运行程序，效果如图 5-20 所示。

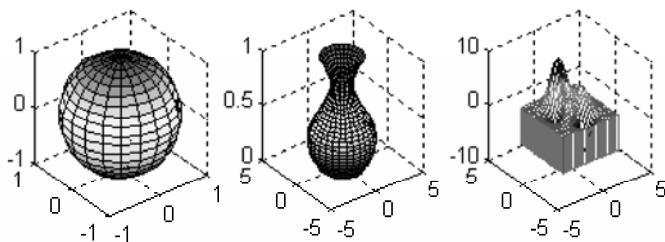


图 5-20 标准三维曲面图

5.2.3 其他三维图形绘制

在介绍二维图形时，曾提到条形图、杆图、饼图和填充图等特殊图形，它们还可以以三维形式出现，使用的函数分别是 `bar3`、`stem3`、`pie3` 和 `fill3`。

它们的绘制格式参看以下示例。

【例 5-18】绘制三维图形：

- (1) 以三维杆图形式绘制曲线 $y = 2\sin(x)$ 。
- (2) 用随机的顶点坐标值画出 5 个黄色三角形。
- (3) 绘制魔方阵的三维条件图。
- (4) 已知 $x = [2014, 1982, 2123, 3177]$ ，绘制三维饼图。

代码如下：

```
subplot(2,2,1);
y=2*sin(0:pi/15:2*pi);
stem3(y);
subplot(2,2,2);
fill3(rand(3,6),rand(3,6),rand(3,6),'y');
subplot(2,2,3);
bar3(magic(4));
subplot(2,2,4);
pie3([2014,1982,2123,3177]);
```

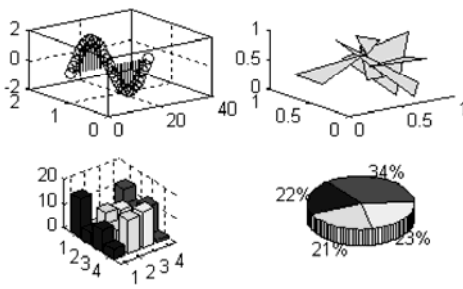


图 5-21 其他三维图形

运行程序，效果如图 5-21 所示。

除了上面讨论的三维图形外，常用图形还有瀑布图和三维曲面的等高线图。绘制瀑布图用 `waterfall` 函数，它的用法及图形效果与 `meshz` 函数相似，只是它的网格线在 x 轴方向出现，具有瀑布效果。等高线图分二维和三维两种形式，分别使用函数 `contour` 和 `contour3` 绘制。调用格式请参看以下示例。

【例 5-19】绘制多峰函数的瀑布图和等高线图。

代码如下：

```
subplot(1,2,1);
[x,y,z]=peaks(30);
waterfall(x,y,z);
xlabel('x-axis');ylabel('y-axis');zlabel('z-axis');
subplot(1,2,2);
contour3(x,y,z,15,'r'); %其中 15 代表高度的等高线
xlabel('x-axis');ylabel('y-axis');zlabel('z-axis');
```

运行程序，效果如图 5-22 所示。

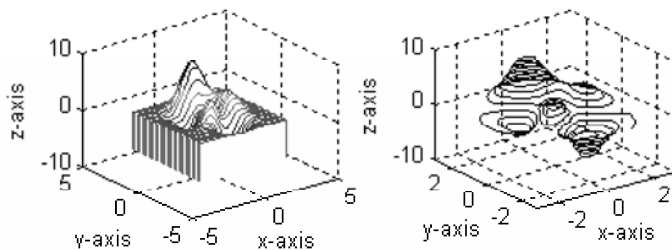


图 5-22 瀑布图和三维等高线图

5.2.4 透明度作图

在 MATLAB 中使用 `hidden` 函数控制移除三维图形中显示的隐藏线。隐藏线的移除是从视图上看由于被其他物体遮住而模糊不清的线。其调用格式如下：

`hidden on`

对当前图形打开隐藏线移除的状态，因此三维图后方的线会被前面的线遮住，简单来说就是透视被叠压的图形。

`hidden off`

对当前图形关闭隐藏移除的状态，因此三维图后方的线将不会被前面的线遮住，也就是说该三维图会变成一个透明的图。

`hidden`

切换 `hidden` 为 `on` 或 `off` 的状态。

【例 5-20】绘制一个三维图形，一个球体，球体包住三维图形，并且将球体设置为透明。

代码如下：

```
[x,y,z]=sphere(25);
x=8.5*x;
y=8.5*y;
```

```

z=8.5*z;
peaks;shading interp; %使用 interp 渲染方式
colormap(hot); %使用 hot 颜色映射值
hold on;
mesh(x,y,z);hold off; %以 mesh 来绘制球体数据
axis equal; %产生长的坐标轴以便于球体的显示
axis off %将坐标轴隐藏

```

输出结果如图 5-23 (a) 所示。现在在以上代码中加入以下语句：

```
hidden off %将球体设置为透明
```

得到的结果如图 5-23 (b) 所示。

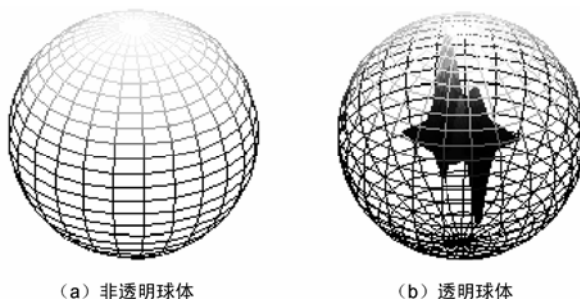


图5-23 透明作图效果

5.2.5 立体可视化

除了前面介绍的常用网格图、表面图和等高线图外，MATLAB 还提供了一些立体可视函数用于绘制更为复杂的立体和向量对象。这些函数通常在三维空间中构建标量和向量的图形。由于这些函数构建的是立体而不是一个简单的表面，因此它们需要三维数组作为输入参数，其中三维数组的每一维分别代表一个坐标轴，三维数组中的点定义了坐标轴栅格和坐标轴上的坐标点。如果要绘制的函数是一个标量函数，则绘图函数需要 4 个三维数组，其中 3 个数组各代表一个坐标轴，第四个数组代表了这些坐标处的标量数据，这些数组通常记为 X 、 Y 、 Z 和 V 。如果要绘制的函数是一个向量函数，则绘图函数需要 6 个三维数组，其中 3 个各表示一个坐标轴，3 个用来表示坐标点处的向量，这些数组通常记为 X 、 Y 、 Z 、 U 、 V 和 W 。

要正确合理地使用 MATLAB 提供的立体和向量可视化函数，用户需要对与立体和向量有关的一些术语有所了解。比如，散度 (Divergence) 和旋度 (Curl) 用于描述向量过程，而等值面 (Isosurfaces) 和等值项 (Isocaps) 则用于描述立体的视觉外观。如果用户要生成和处理比较复杂的立体对象，就需要参考相应的文献对这些术语进行深入了解。在此并不详细介绍这些术语的含义，只通过以下几个示例进行应用。

【例 5-21】利用标量函数构建立体图形示例。

代码如下：

```

>> x=linspace(-4,4,16);
y=1:25;
z=-5:5;
[X,Y,Z]=meshgrid(x,y,z);
size(X)

```

显示如下：

```
ans =  
    25    16    11
```

上面的代码演示了 `meshgrid` 函数在三维空间中的应用。其中， X 、 Y 、 Z 为定义栅格的 3 个三维数组。这 3 个数组分别是 x 、 y 和 z 经过三维栅格扩展形成的。我们需要定义一个以这三个数组为自变量的标量函数 V ，代码如下：

```
>> V=sqrt(X.^2+cos(Y).^2+Z.^2);
```

这样，利用标量函数 $v = f(x, y, z)$ 定义一个立体对象所需要的数据已全部给出。为了使该对象可视化，可以利用下面的代码查看该立体对象的一些截面。

```
slice(X,Y,Z,V,[0,4],[5,15],[-4 6])  
xlabel('X-axis');  
ylabel('Y-axis');
```

运行程序，效果如图 5-24 所示。

除了查看立体对象的截面之外，寻找使 V 等于某个特定值的表面（称为等值面）也十分常见。在 MATLAB 中，这一操作可以用 `isosurface` 函数来实现，该函数与 `delaunay` 函数类似，由这些三角形构成等值面。下面给出一个绘制等值面的示例。

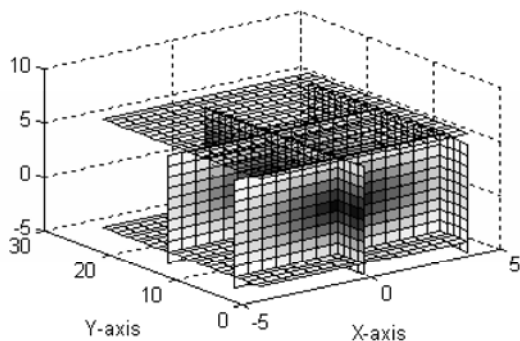


图 5-24 立体截面图

【例 5-22】绘制等值面示例。

代码如下：

```
[X,Y,Z,V]=flow(13);  
f1=isosurface(X,Y,Z,V,-2);  
subplot(1,2,1);  
p=patch(f1);  
set(p,'FaceColor',[0.5 0.5 0.5],'EdgeColor','Black'); %设置面属性  
view(3);  
axis equal; grid on; %按比例显示图形，打开网格  
subplot(1,2,2);  
p=patch(shrinkfaces(f1,0.3));  
set(p,'FaceColor',[0.5 0.5 0.5],'EdgeColor','Black');  
view(3);  
axis equal; grid on; %按比例显示图形，打开网格
```

运行程序，效果如图 5-25 所示。

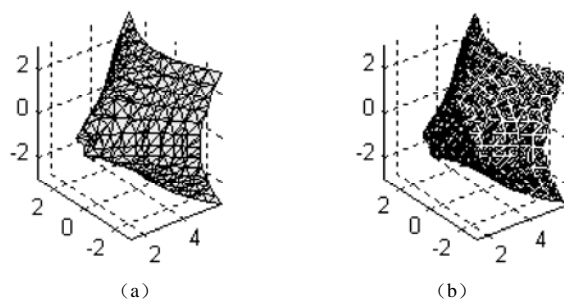


图 5-25 三维等值图

图 5-25 (b) 还展示了函数 `shrinkfaces` 的用法，顾名思义，该函数的功能是使表面收缩。三维数据也可以通过用 `smooth3` 函数来过滤而实现其平滑化，看以下示例。

【例 5-23】`smooth3` 函数用法。

代码如下：

```
clear;
data1=rand(12,12,12);
data2=smooth3(data1,'box',3);
subplot(1,2,1);
p=patch(isosurface(data1,0.5),'FaceColor','Blue','EdgeColor','none');
patch(isocaps(data1,0.5),'FaceColor','interp','EdgeColor','none');
isonormals(data1,p);
view(3);
axis vis3d tight off;
camlight;lighting phong;
subplot(1,2,2);
p=patch(isosurface(data2,0.5),'FaceColor','Blue','EdgeColor','none');
patch(isocaps(data2,0.5),'FaceColor','interp','EdgeColor','none');
isonormals(data2,p);
view(3);
axis vis3d tight off; %设置坐标轴比例因子相等
camlight;lighting phong; %生成摄像机函数并将其放在合适的位置
```

运行程序，效果如图 5-26 所示。

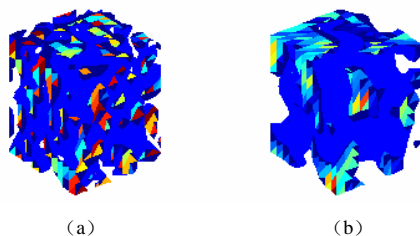
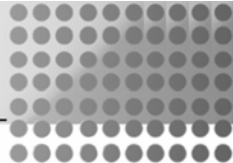


图 5-26 三维数据平滑化

上边的例子展示了函数 `isocaps` 和 `isonormals` 的用法。函数 `isocaps` 生成块状图的外层表面。函数 `isonormals` 调整所画碎片的属性，使得所显示的图形有正确的光照效果。



5.3 图形颜色映像的应用

在图形表示过程中,颜色的运用能反映出许多其他的图形信息,所以颜色映像运用也是一个十分重要的环节。

1. 基本的着色技术

由于色彩在表现图形中非常重要,所以 MATLAB 特别重视色彩处理,而色图是 MATLAB 着色的基础。

语句 `colormap(M)` 将矩阵 **M** 当成当前图形窗口所用的颜色映像。例如, `colormap(cool)` 装入了一个有 64 个输入项的 cool 颜色映像。`colormap default` 装入了默认的颜色映像 `hsv`。其他颜色映像请读者参看联机帮助文档。在此不再介绍。

函数 `colormap` 的调用格式如下:

```
colormap(MAP)
```

该函数用于把当前图形的颜色映像设为 **MAP**, **MAP** 可以是 MATLAB 提供的颜色映像,如“jet”,也可以自己定义颜色映像矩阵。需要注意的是,矩阵 **MAP** 的行数不限,但必须为 3 列。

至此,我们一直在讲颜色映像,那么这些映像对应的到底是怎样的颜色呢?我们没有一个直观的认识,下面就来介绍能将颜色映像直观显示的函数。

2. 颜色映像的直观显示

颜色映像的直观显示有如下几个知识点。

(1) 观察颜色映像矩阵元素

可以通过多个途径来显示一个颜色映像,其中一个方法就是观察颜色映像矩阵的元素。在 MATLAB 命令窗口输入如下命令:

```
>> hot(8)
```

显示如下:

```
ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
```

上面的数据显示出第 1 行是 1/3 红色,而最后一行是白色。

(2) `rgbplot` 函数

函数 `rgbplot` 直接把颜色映像矩阵中的 3 列数分别用红、绿、蓝 3 种颜色画出来,例如:

```
>> rgbplot(hot)
```

效果如图 5-27 所示。

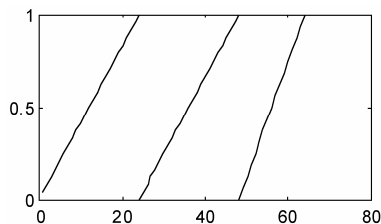


图 5-27 颜色映像 hot 的 RGB 曲线



上面绘制的就是颜色映像 **hot** 矩阵中的 **RGB** 数据，图中的红色曲线（左边）对应着矩阵的第 1 列；绿色曲线（中间）对应着矩阵的第 2 列；蓝色曲线（右边）对应着矩阵的第 3 列。从图中可以分析颜色的变换过程，图中从左到右表示颜色映像从开始到结束的变化，从下向上表示颜色的强度由小到大。开始 3 种颜色都是 0 或接近 0，因而是黑色；之后红色增强，而绿色和蓝色仍为 0，这一段为红色；红色到 1 后，绿色开始增强，蓝色仍然为 0，红色和绿色逐渐合成黄色；绿色达到 1 后，蓝色逐渐增强，当蓝色最后也达到 1 后，3 种颜色合成白色。

（3）**pcolor**函数

函数 **pcolor** 用于绘制伪彩色图。所谓伪彩色是指绘图使用的色彩用于表示数据的大小，而不是自然的色彩。函数 **pcolor** 也可以用来显示一个颜色映像，具体调用格式如下：

pcolor(c)

该函数的作用是将矩阵 **c** 作为颜色矩阵，利用着色原理在平面网格点 (i, j) 的右上角小区域内用 $c(i, j)$ 对应的色谱矩阵的颜色着色。绘制后的伪彩色图还可以用 **shading** 函数调整其颜色的平滑度。例如：

```
>> pcolor(cool(20))
```

输出结果如图 5-28 所示。

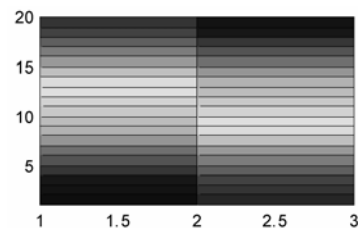


图 5-28 颜色映像 **cool** 的伪彩色图

（4）**colorbar**函数

函数 **colorbar** 在当前的图形窗口中增加水平或者垂直的颜色标尺以显示当前坐标轴的颜色映像。该函数的用法如下：

colorbar

如果当前没有颜色条就加一个垂直的颜色条，或者更新现有的颜色条。

colorbar('horiz')

在当前的图形下面放一个水平的颜色条。

colorbar('vert')

在当前的图形右边放一个垂直的颜色条。

3. 颜色映像的建立和修改

颜色映像就是矩阵，意味着用户可以像其他数组那样对它们进行操作。**MATLAB** 提供了一系列的函数建立和修改颜色映像矩阵，如表 5-2 所示。

表 5-2 建立和修改颜色映像矩阵函数

函数名称	描 述
brighten	通过调整一个给定的颜色映像来增加或者减少暗色的强度
caxis([cmin,cmax])	用于设置伪彩色的缩放比例。其中， cmin 和 cmax 分别表示当前颜色映像中第一个颜色和最后一个颜色对应的数据大小

【例 5-24】表 5-2 所示的两个函数示例。

代码如下：

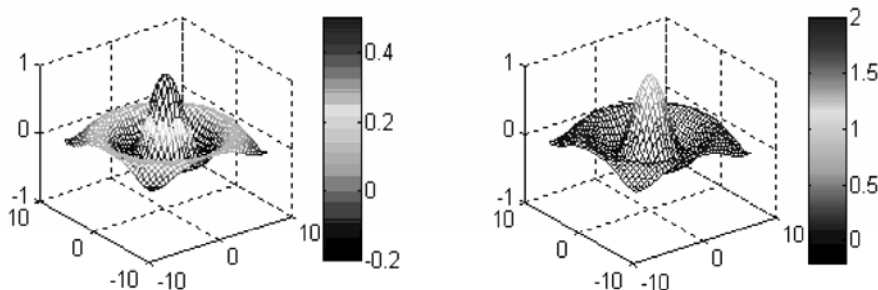
```
[x,y]=meshgrid(-8:0.5:8);
r=sqrt(x.^2+y.^2)+eps;
```

```
z=sin(r)./r;
mesh(x,y,z);
caxis([-0.2,0.5]);
colorbar('vert');
```

输出结果如图 5-29 (a) 所示。

```
>> caxis([-0.2,2])
>> colorbar('vert')
```

输出结果如图 5-29 (b) 所示。



(a) 颜色映像的范围小于数据范围

(b) 颜色映像的范围超出了数据范围

图 5-29 建立和修改颜色映像矩阵函数示例

除此之外，用户还可以自定义颜色映像。

可以通过生成 $m \times 3$ 的矩阵 `mymap` 来建立用户的颜色映像，并用 `colormap(mymap)` 来安装它。颜色映像矩阵的每一个值都必须在 0 和 1 之间。如果企图用大于或小于 3 列的矩阵或者包含着比 0 小或者比 1 大的任意值，函数 `colormap` 会提示一个错误信息，然后退出。也可以组合颜色映像，其结果有时是不可预料的。只有当所有元素都在 0 与 1 之间时，才能保证结果是一个有效的颜色映像。

5.4 光照和材质处理

5.4.1 光照处理

使用光照处理能够把图形表现得更加逼真，得到非常真实的视觉效果。为了创建光照效果，MATLAB 提供了光源 (Light) 图形对象。用 `light` 函数可以创建 Light 对象，该函数的调用格式为：

```
light('Color',选项 1,'Style',选项 2,'Position',选项 3)
```

Light 对象有 3 个重要的属性。其中 Color 属性确定光的颜色，选项 1 取 RGB 三元组或相应的颜色字符。Style 属性确定光源的类型，选项 2 有 `infinite` 和 `local` 两个取值，分别表示无穷远光和近光。Position 属性指定光源的位置，选项 3 取三维坐标点组成的向量形式 `[x,y,z]`。对无穷远光，它表示穿过该点射向原点，对于近光，它表示光源所在位置。假如函数不包含任何参数，则采用默认设置：白光、无穷远、穿过 (1, 0, 1) 点射向坐标原点。

利用 `lighting` 命令可以设置光照模式，其格式为：

```
lighting 选项
```

其中，选项有 4 种取值，flat、gouraud、phong、none。flat 选项使得入射光均匀洒落在图形对象的每个面上，是默认选项；gouraud 选项先对顶点颜色插补，再对顶点勾画的面上颜色进行插补，用于表现曲面；phong 选项对顶点处的法线插值，再计算各个像素的反光，它生成的光照效果好，但费时；none 选项关闭所有光源。

【例 5-25】绘制光照处理后的球面并观察不同光照模式下的效果。

代码如下：

```
[x,y,z]=sphere(20);
subplot(1,4,1);
surf(x,y,z);
axis equal;shading interp;
hold on;
subplot(1,4,2);
surf(x,y,z);axis equal;
light('Position',[0,1,1]);
shading interp;lighting flat;
hold on;
plot3(0,1,1,'p');
text(0,1,1,'light');
subplot(1,4,3);
surf(x,y,z);axis equal;
light('Position',[0,1,1]);
shading interp;
lighting gouraud;
hold on;
subplot(1,4,4);
surf(x,y,z);axis equal;
shading interp;
lighting phong;
```

程序执行结果如图 5-30 所示。图中第一至第四个球分别是没有使用光照、使用 flat 光照、使用 gouraud 光照和使用 phong 光照时的显示效果，第二个球还标出光源的位置。

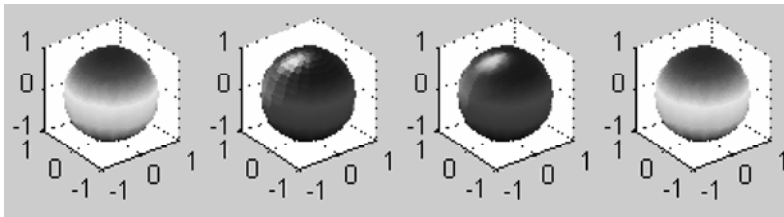
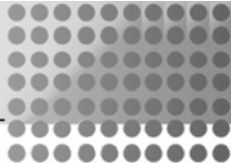


图 5-30 光照处理后的图形

5.4.2 材质处理

材质体现了图形对象的反射特性，修改区域块和曲面对象的反射特性，可以改变在场景中应用光照时对象的显示外观。这些特性包括：镜面反射和漫反射、环境光、镜面反射指数、镜面反射光的颜色和背面光照。可以组合使用这几种特性来生成特殊的显示效果。



1. 镜面反射和漫反射

区域块和曲面对象的 **SpecularStrength** 属性用来控制对象表面镜面反射的强度, 属性值取 0~1 之间的数, 默认值为 0.9。**DiffuseStrength** 属性用来控制对象表面漫反射的强度, 属性值取 0~1 之间的数, 默认值为 0.6。

2. 环境光

环境光不是镜面光, 它均匀地洒在场景中的所有对象上。只有在坐标系中有 **Light** 对象时环境光才可见。**AmbientStrength** 属性是一个用于区域块和曲面对象的属性, 确定特定对象上环境光的强度, 属性值取 0~1 之间的数, 默认值为 0.3。

3. 镜面反射指数

镜面反射光的大小与区域块和曲面对象的 **SpecularExponent** 属性有关, 该属性的值介于 1~500 之间, 默认值为 10。

4. 镜面反射光的颜色

镜面反射光的颜色可以有一个变化范围, 即从对象颜色与光源颜色的组合色变化到只有光源颜色。区域块和曲面对象的 **SpecularColorReflectance** 属性控制这个颜色, 属性值取 0~1 之间的数, 默认值为 1。

5. 背面光照

背面光照可用于显示对象内表面和外表面的差别。区域块和曲面对象的 **BackFaceLightig** 属性控制该效果, 属性取值为 **unlit**、**lit** 和 **reverselit** (默认)。

6. material 函数

使用 **material** 函数也可以设置区域块和曲面对象的表面反射特性。该函数的调用格式有如下几种。

- **material shiny**: 镜面反射光的强度比漫反射光和环境光的强度要高得多, 镜面光的颜色只取决于光源的颜色。
- **material dull**: 主要进行漫反射, 没有镜面反射, 但是反射光的颜色只与光源有关。
- **material metal**: 镜面反射很强, 环境光和漫反射光较弱, 反射光的颜色与光源和对象的颜色都有关系。
- **material([ka kd ks])**: 设置对象的环境光、漫反射光和镜面光的强度。
- **material([ka kd ks n])**: 设置环境光、漫反射光、镜面光的强度以及对象的镜面反射指数。
- **material([ka,kd,ks,n,sc])**: 设置环境光、漫反射光、镜面光的强度、对象的镜面反射指数及镜面反射光的颜色。
- **material default**: 将环境光、漫反射光、镜面光的强度、对象的镜面反射指数和镜面反射光的颜色设置为默认值。



material 命令设置坐标系中所有区域块和曲面对象的 **AmbientStrength**、**DiffuseStrength**、**SpecularStrength**、**SpecularExponent** 和 **SpecularColorReflectance** 属性。坐标系中必须有一个可见的 **Light** 对象。

【例 5-26】生成一个球体和一个立方体, 观察不同光照属性对应的显示效果。
代码如下:



```

sphere(35);
h=findobj('Type','surface');
set(h,'FaceLighting','phong','FaceColor','interp',...
    'EdgeColor',[0.4 0.4 0.4],'BackFaceLighting','lit');
hold on;
vert=[2 0 -1;2 1 -1;3 0 0;3 0 -1;2 0 0;2 1 0;3 1 0;3 0 0];
fac=[1 2 3 4;2 6 7 3;4 3 7 8;1 5 8 4;1 2 6 5;5 6 7 8];
patch('Faces',fac,'Vertices',vert,'FaceColor','r');
light('Position',[1 3 2]);
light('Position',[-3 -1 3]);
material shiny;
axis equal;
hold off

```

程序中用 `findobj` 函数查找 `Type` 属性为 `surface` 的对象，从而可以获取该球面的句柄，进而设置其属性。球面使用了 `phong FaceLighting` 属性值，因此生成了最平滑的光照插值效果。`vert` 和 `fac` 定义立方体。默认时，立方体使用 `flat Facelighting` 属性值增强每个边的可见性。`material shiny` 命令会影响立方体和球体的反射属性。因为球体是闭合的，所以 `BackFaceLighting` 属性从默认设置变成了正常光照，删除了不必要的边缘效应。程序运行后，生成的图形效果如图 5-31 所示。

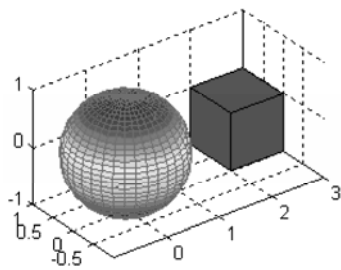


图 5-31 光照属性设置效果

5.5 图像显示技术

5.5.1 图像简介

在 MATLAB 中，图像（Image）通常由数据矩阵和色彩矩阵组成。根据图像着色方法的不同，MATLAB 图像可以分为 3 类：索引图像（Indexed Image）、亮度图像（Intensity Image）和真彩色图像（True Color or RGB Image）。

索引图像是带有颜色表矩阵的，图像数据矩阵中的数据通常被解释成指向颜色表的矩阵的索引号。图像颜色表矩阵可以是上一节讲到的任何有效的颜色表：即任何包含了有效 RGB 数据的 $m \times 3$ 的数组。如果索引图像的图像数据数组为 $X(i, j)$ ，颜色表数组为 `cmap`，则每个图像像素 $P_{i,j}$ 的颜色就是 `cmap(X(i,j),:)`。这要求 X 中的数据值必须是位于 $[1 \text{ length}(\text{cmap})]$ 范围内的整数。如果用户已经获得图像数据和颜色表，可以使用下面的命令显示这幅图像：

```
>> image(X);colormap(cmap)
```

亮度图像的图像数据矩阵通常表示该图像的亮度值。该类型图像通常用于显示由灰度或单色颜色表染色的图像，有时也用于其他颜色表染色的图像。亮度图像对数据范围没有要求，不一定要像索引图像那样位于 $[1 \text{ length}(\text{cmap})]$ 范围之内。用户可以指定亮度图像的数据范围，并且将其作为指向颜色表的索引。如下面的例子：

```
>> image(X,[0 1]);colormap(gray)
```

将 X 的值限制在 $[0 \ 1]$ 之间，并将 0 指向颜色表的第一个颜色，将 1 指向颜色表的最后一个

颜色，介于 0 和 1 之间的数据被用来作为指向颜色表中其他颜色的索引。如果上面的语句省略 [0 1]，则意味着不对 X 进行限定，也就是说， X 的数据范围是 $[\min(\min(X)) \max(\max(X))]$ 。

真彩色（也叫 RGB）图像通常由一个包含有效 RGB 值的 $m \times n \times 3$ 的数组创建。该数组的行和列声明了像素的位置，也声明了图像中每一个像素的颜色值。也就是说，像素 $P_{i,j}$ 将用 $X(i,j,:)$ 所声明的颜色绘制。由于真彩色图像已经将颜色信息包含在图像数据中，因此它不需要颜色表。如果计算机硬件不支持真彩色图像（例如，它只有一块 8 位显卡），那么 MATLAB 就利用颜色近似和抖动来显示图像。真彩色图像的显示比较简单，如下例所示：

```
>> image(X)
```

其中， X 是一个 $m \times n \times 3$ 的真彩色图像。 X 可以包含双精度数据，也可以包含 unit8、unit16 类型的数据。

如果事先不知道图像类别，那么就先使用 `imfinfo` 指令获取该图像的信息，然后再进行读操作。图像着色类型不同，其显示和写入指令也不同。以下命令可以用来获取图像文件的特征数据（特别是着色类型 `ColorType`）。

```
>> imfinfo(fileName)
```

指令 `imfinfo` 将产生一个构架数组。不管数组的大小如何，在构架上都有一个名为 `ColorType` 的域，域中存放着如下 3 种“图像着色类型字符串”。

- `indexed`：变址着色的图像。
- `grayscale`：灰度着色的图像。
- `truecolor`：真彩着色的图像。

`Parameter/Value` 用来修改对象属性。常用的 `Parameter/Value` 随图像格式不同而不同，具体情况如表 5-3 所示。

表 5-3 常用的 `Parameter/Value`

格 式	Parameter	Value	默 认 值
JPEG	Quality	[0, 100]之间的任何数	75
TIFF	Compression	none, packbits 对二维图像可选 ccitt	二维图像用 ccitt，其余用 packbits
	Description	任何字符串	空串
HDF	Compression	none, rle, jpeg	rle
	WriteMode	overwrite, append	overwrite
	Quality	[0,100]之间的任何数	75

5.5.2 图像的读取

不同的类型图像有自己固定的数据格式。要在 MATLAB 中使用其他软件中使用的图像，需要用 `imread` 函数读取该图像。这实际上也是一个数据转换的过程，即把该图像的数据转换成 MATLAB 图像的数据格式。函数 `imread` 的调用格式如下：

- `A=imread(filename,fmt)`：返回存放图像的变量名 A 。
- `[X, MAP]=imread(filename, fmt)`：返回图像的数值存放矩阵 X 和颜色矩阵 MAP 。
- `[...]=imread(filename)`：返回图像的信息。

其中，`filename` 为图像的文件名；`fmt` 指定图像的类型，可以为 JPEG (`jpg` 或 `jpeg`)、TIFF (`tif` 或者 `tiff`)、BMP (`bmp`)、PNG (`png`)、HDF (`hdf`)、PCX (`pcx`) 和 XWD (`xwd`)； A 为图像文

件中读出并转化成 MATLAB 可识别的图像格式的数据；X 为保存索引图像数据的数组；MAP 为保存相关颜色映像的数组。

【例 5-27】imread 函数应用示例。

```
>> A=imread('link.jpg','jpg');
size(A)
ans =
    104    139     3
>> A=imread('link','jpg');
>> size(A)
ans =
    104    139     3
>> A=imread('link.jpg');
>> size(A)
ans =
    104    139     3
```

可以通过上述的 3 种方式来读取真彩色文件“link.jpg”。读者可以看到三维数组“A”有 3 个面，它依次为 R、G、B 3 种颜色，而面上的数据则分别是这 3 种颜色的强度值，面中的元素对应于图像中的像素点，因而面中的行数和列数与图像中像素的行数和列数是一致的。

MATLAB 中的函数 imwrite 用于把图像输出到文件，调用格式如下：

- imwrite(A, filename, fmt)：将变量 A 以 fmt 格式存为 filename。
- imwrite(..., filename)：将当前图像矩阵以文件名 filename 存储。
- imwrite(..., 'ProName','ProVal',...): 根据属性名 ProName 的值另存图像数据。

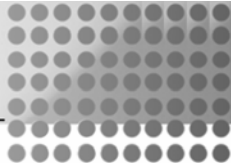
其中，参数 filename 和 fmt 与函数 imread 相同。而在第三条命令中，参数随着 fmt 的改变而改变，具体请参阅 MATLAB 帮助文件。

MATLAB 支持的一些常用图像与图形格式如表 5-4 所示。

表 5-4 MATLAB 支持的一些常用图像与格式

格式名称	描 述	可识别扩展名
JPEG	联合图像专家组	.jpg、.jpeg
TIFF	加标识的图像文件格式	.tif、.tiff
PNG	可移植的网络图形	.png
GIF	图形交换格式	.gif
BMP	Windows 位图	.bmp
HDF	面向对象的自描述图像格式	.hdf
XWD	X Window 转储	.xwd
ICO	图标资源文件	.ico
CUR	光标资源文件	.cur
RAS	光栅图像位图	.ras
PCX	Windows 画刷图形	.pcx
PGM	简便灰度图像	.pgm
PBM	简便位图格式	.pbm
PPM	简便像素图形	.ppm

对于上表中的 GIF 格式图像，imread 函数支持，但是 imwrite 函数不支持。



5.5.3 图像的显示

众所周知，数字图像是指将一幅二维的图像表示成一个数值矩阵，矩阵的元素被解释为像素的颜色值（或灰度值），或被解释为调色板颜色的索引号。为了显示由矩阵表示的数字图像，MATLAB 最一般的做法是将矩阵的每个元素与当前色谱的某个行标号对应，并取出该行的颜色值作为图像相应点的颜色。一般来说，每幅图的色调不同，因此作为图像必须有自己特殊的色图，这样才能真实地显示图像。

MATLAB 用函数 `image` 显示图像，其调用格式如下：

- `image(C)`: 把矩阵 `C` 作为一个图像画出。
- `image(x,y,C)`: 在 (x,y) 确定的位置上画 `C` 的元素。
- `image(x,y,C,'ProName','ProValue',...)`: 指定属性名和属性值，在 (x,y) 确定的位置上画 `C` 的元素。
- `image('ProName','ProValue',...)`: 只接受属性名和属性值的输入。
- `h=image(...)`: 返回刚生成的图片对象的句柄属性值向量。

另一个与 `image` 函数类似的函数是 `imagesc`，它的命令格式与 `image` 一样。`image(X)` 是将数据矩阵 `X` 的值直接作为索引号在色谱矩阵中提取 RGB 颜色值进行着色的。事实上，对于任何矩阵 `X`，`image(X)` 可以生成一幅图像。如果 `X` 元素的数值大小十分接近，或超出色谱矩阵的长度，那么 `image(X)` 就不能有效地用图像表达矩阵 `X`，而函数 `imagesc` 就可以做到这一点。`Imagesc` 的功能与 `image` 是一样的，只是按线性变换的方式计算索引号，即与 `pcolor` 使用的方法相同。于是，`imagesc(X)` 生成的图像将受到 `caxis` 函数的影响。

【例 5-28】在根目录下有一图像文件 `moon.jpg`，在图形窗口显示该图像。

代码如下：

```
[x,cmap]=imread('moon.jpg'); %读取图像的数据阵和色图阵
image(x);                    %显示图像
colormap(cmap);
axis image off;              %保持宽高比并取消坐标轴
```

运行程序，效果如图 5-32 所示。



图 5-32 图像显示

5.6 动画制作技术

有两种常见的动画形式，一种是影片动画，影片动画预先制作图形，存储在图形缓冲区，然后逐帧播放。这种动画形式适用于难以实时绘制的复杂画面。这种方法计算量大，占用内存多，但回放速度快，画面连贯。另一种是实时动画。实时动画保持图形窗口中绝大部分的像素色彩不变，而只是更新部分像素的颜色从而构成运动图像。这种动画形式适用于每次变化较少、图形精度变化不是很高的场合。

1. 影片动画制作

如果将 MATLAB 产生的多幅图形保存起来，并利用系统提供的函数进行播放，就可以产生动画效果。MATLAB 提供了 3 个函数用于捕捉和播放动画，它们分别为 `getframe`、`moviein` 和 `moive`。

`getframe` 函数可截取每一幅画面信息而形成很大的列向量。该向量可保存到一个变量中。显然，保存多幅图就需一个大矩阵。



moviein(n)函数用来建立一个足够大的 n 列矩阵。该矩阵用来保存 n 幅画面的数据,以备播放。之所以要事先建立一个矩阵,是为了提高程序运行速度。

moive(m,n)函数播放由矩阵 m 所定义的画面 n 次,默认时播放一次。

【例 5-29】播放一个直径不断变化的球体。

代码如下:

```
[x,y,z]=sphere(45);
m=moviein(32); %建立一个 32 列大矩阵
for i=1:32
    surf(i*x,i*y,i*z); %绘制球面
    m(:,i)=getframe; %将球面保存到 m 矩阵
end
movie(m,11); %播放球面 11 次
```

2. 实时动画制作

制作实时动画的基本方法是:先画出初始图形,再计算活动对象的新位置,并在新位置上把它显示出来,最后擦除原位置上原有的对象,刷新屏幕。重复操作即可产生动画效果。

在 MATLAB 中,利用图形的 EraseMode 属性,可以实现 3 种重要的擦除方式。

① None: 在图形对象变化时,不做任何擦除而直接在原来图形上绘制。

② Background: 在图形对象被擦除后, MATLAB 将原来对象的颜色设为背景颜色,实现擦除。这种模式将原来的图形对象完全擦除,包括该对象下面的所有对象。

③ Xor: 对象的绘制和擦除由该对象颜色与屏幕颜色的异或而定。只绘制与屏幕颜色不一样的新对象点。这种模式只擦除与屏幕颜色不一样的原对象点,而不损害被擦除对象下面的其他对象。大多数 MATLAB 动画都采用这种擦除方式。

当新对象属性设置后,应该及时刷新屏幕,从而使新对象显示出来。这些操作依靠命令 drawnow 完成。drawnow 命令迫使 MATLAB 暂停目前的任务序列而去刷新屏幕。若没有 drawnow 命令, MATLAB 要等任务序列执行完后才去刷新屏幕。一般说来,在实时动画中,为更新屏幕,执行 drawnow 命令是必需的。

【例 5-30】模拟布朗运动。

代码如下:

```
n=35; %指定布朗运动的点数
a=0.02; %指定温度
%产生 n 个随机点 (x,y), 处于-0.5 到 0.5 之间
x=rand(n,1)-0.5;
y=rand(n,1)-0.5;
h=plot(x,y,'.'); %绘制随机点
axis([-1 1 -1 1]);
axis square;
grid off;
set(h,'EraseMode','Xor','MarkerSize',24); %设置擦除模式为 Xor
%循环 4900 次,产生动画效果
for i=linspace(1,10,4900)
    drawnow
    x=x+a*randn(n,1); %在坐标点附近添加随机噪声
    y=y+a*randn(n,1);
    set(h,'xdata',x,'ydata',y); %通过改变数据属性来重新绘图
end
```

运行程序,效果如图 5-33 所示。

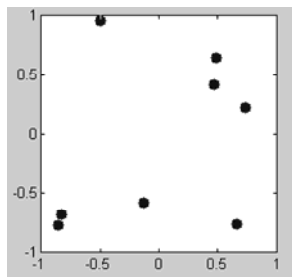


图 5-33 实时动画效果

第6章 MATLAB在模糊控制系统中的应用

6.1 模糊系统的MATLAB实现

6.1.1 模糊集简介

1. 模糊概念

在现实世界中,有很多事物的分类边界是不分明的或者说是难以明确划分的。如人的高矮、胖瘦,“温度偏离”、“压力偏大”等,为了用数学方法描述这类概念,引入模糊集合。

模糊集是一种边界不分明的集合,模糊集与普通集合既有区别又有联系。对于普通集合而言,任何一个元素要么属于该集合,要么不属于该集合,非此即彼,具有精确明了的边界;而对于模糊集合,一个元素可以既属于该集合又不属于该集合,亦此亦彼,边界不分明或界限模糊。

建立在模糊集基础上的模糊逻辑,任何陈述或命题的真实性只是一定程序的真实性,与建立在普通集合基础上的布尔逻辑相比,模糊逻辑是一种广义化的逻辑。在布尔逻辑中,任何陈述或命题只有两种取值,即逻辑真和逻辑假,常用“1”表示逻辑真,“0”表示逻辑假。而在模糊逻辑中,陈述或命题的取值除真和假(“1”和“0”)外,可取0~1之间的任何值,如0.75,即命题或陈述在多大程度上为弄虚作假,例如“老人”这一概念,在普通集合中没有一个明确的边界,60岁以上是老人,58岁也属于老人,40岁在一定程度上也属于老人,只是他们属于老人这一集合的程度不同。模糊性反映了事件的不确定性,但这种不确定性不同于随机性。随机性反映的是客观性,即人们对有关事件定义或概念描述在语言意义理解上的不确定性。

2. 模糊集的表达

模糊集采用隶属度来表示,论域 X 上的一个模糊集 A ,对于任意 $x \in X$,都指定了一个数 $\mu_A(x) \in [0, 1]$,叫做 x 对 A 的隶属程度,称为 A 的隶属函数。

论域 X 中的模糊集 A 的3种表示方法如下。

- 如果 X 是有限集或可数集,则 A 可表示为: $A = \sum_{x_i} \frac{\mu_A(x_i)}{x_i}, (x_i \in X)$ 。

如果 X 是无限不可数集, A 也可表示为: $A = \int_X \frac{\mu_A(x)}{x} dx$ 。

- 向量表示法: $X = \{x_1, x_2, \dots, x_n\}$; $A = [\mu_A(x_1), \mu_A(x_2), \dots, \mu_A(x_n)]$ 。
- 序偶表示法: $A = \{x, \mu_A(x) | x \in X\}$ 。

模糊集使得某元素可以以一定程度属于某集合,某元素属于某集合的程度由“0”与“1”之间的一个数值来刻画或描述。把一个具体的元素映射到一个合适的隶属度是由隶属度函数来实现的。隶属度函数可以是任意形状的曲线,取什么形状取决于是否使用起来感到简单、方便、快速、有效,唯一的约束条件是隶属度函数的值域为 $[0, 1]$,模糊系统中常用的隶属度函数有如下11种。

(1) 高斯型隶属度函数

$$f(x, \sigma, c) = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

高斯型隶属度函数有 2 个特征参数 σ 和 c 。

(2) 双侧高斯型隶属度函数

双侧高斯型隶属度函数是两个高斯型隶属度函数的组合, 有 4 个参数 σ_1 、 c_1 、 σ_2 、 c_2 。 c_1 与 c_2 之间的隶属度为 1, c_1 左边的隶属度函数为高斯型隶属度函数 $f(x, \sigma_1, c_1)$, c_2 右边的隶属度函数为高斯型隶属度函数 $f(x, \sigma_2, c_2)$ 。

(3) 钟型隶属度函数

$$f(x, a, b, c) = \frac{1}{1 + \left(\frac{x-c}{a}\right)^{2b}}$$

钟型隶属度函数的形状如钟, 故名钟型隶属度函数, 钟型隶属度函数有 3 个参数 a 、 b 、 c 。

(4) sigmoid 型隶属度函数

$$f(x, a, b, c) = \frac{1}{1 + e^{-a(x-c)}}$$

sigmoid 型隶属度函数有 3 个特征参数 a 、 b 和 c 。

(5) 差型 sigmoid 隶属度函数

$$f(x, a_1, c_1, a_2, c_2) = \frac{1}{1 + e^{-a_1(x-c_1)}} - \frac{1}{1 + e^{-a_2(x-c_2)}}$$

差型 sigmoid 隶属度函数为两个 sigmoid 型隶属度函数之差, 它有 4 个特征参数 a_1 、 c_1 、 a_2 、 c_2 。

(6) 积型 sigmoid 隶属度函数

积型 sigmoid 隶属度函数为两个 sigmoid 型隶属度函数的乘积。

$$f(x, a_1, c_1, a_2, c_2) = \frac{1}{1 + e^{-a_1(x-c_1)}} \cdot \frac{1}{1 + e^{-a_2(x-c_2)}}$$

积型 sigmoid 隶属度函数有 4 个参数 a_1 、 c_1 、 a_2 、 c_2 。

(7) Z 型隶属度函数

Z 型隶属度函数有两个参数 a 、 b , 分别为隶属度函数曲线中斜线部分极点的位置。

(8) S 型隶属度函数

S 型隶属度函数有两个参数 a 和 b , a 、 b 是隶属度函数曲线中斜线部分极点的位置。

(9) II 型隶属度函数

II 型隶属度函数有 4 个参数 a 、 b 、 c 、 d , II 型隶属度函数可以看成参数为 a 、 b 的 S 型函数与参数为 c 、 d 的 Z 型函数叠加而成的。

(10) 梯形隶属度函数

$$f(x,a,b,c,d)=\begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{c-x}{c-b}, & c \leq x \leq d \\ 0, & x \geq d \end{cases}$$

或

$$f(x,a,b,c,d)=\max\left(\min\left(\frac{x-a}{b-a},1,\frac{d-x}{d-c}\right),0\right)$$

梯形隶属度函数有4个参数 a 、 b 、 c 、 d 。

(11) 三角形隶属度函数

$$f(x,a,b,c)=\begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & x \geq c \end{cases}$$

三角形隶属度函数有3个参数 a 、 b 和 c 。

3. 模糊逻辑运算

① 在普通逻辑或布尔逻辑中,任何陈述只有两个取值:真或假(“1”或“0”),即普通逻辑为二值逻辑。二值逻辑关系有逻辑与、逻辑或、直积,对应的逻辑运算有与(交)运算、或(并运算)、非运算、直积等。集合 A 和 B 的二值逻辑运算如下。

- 与运算: $A \cap B = \{x: x \in A \text{ 且 } x \in B\}$ 。
- 或运算: $A \cup B = \{x: x \in A \text{ 或 } x \in B\}$ 。
- 非运算: $\bar{A} = \{x: x \notin A, x \in U, U \text{ 为全集}\}$ 。
- 直积: $A \times B = \{i(a,b): a \in A, b \in B\}$ 。

② 模糊逻辑是普通二值逻辑的推广,在模糊逻辑中,任何陈述都以一定程度的真实性表示,其取值可以是“0”和“1”之间的任意实数,对应的模糊逻辑运算(逻辑与、逻辑或、逻辑非)如下。

- 逻辑与 ($A \text{ AND } B$), $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$ 。
- 逻辑或 ($A \text{ OR } B$), $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$ 。
- 逻辑非 ($\text{NOT } A$), $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$ 。

③ 逻辑与运算和逻辑或运算还可由更广义的模糊逻辑算子——T算子和协T算子来定义。模糊逻辑与运算可由T算子 \otimes 定义为

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \otimes \mu_B(x)$$

T 算子 \otimes 是满足下列条件的一个两变量函数 $T(\cdot, \cdot)$ 。

- 单调：如果 $a \leq c$ 且 $b \leq d$ ，则 $T(a, b) \leq T(c, d)$ 。
- 右界： $T(0, 0) = 0$ ， $T(a, 1) = T(1, a) = a$ 。
- 交换律： $T(a, b) = T(b, a)$ 。
- 结合律： $T(a, T(b, c)) = T(T(b, a), c)$ 。

模糊逻辑或运算也可由协 T 算子 \oplus 定义为

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) = \mu_A(x) \oplus \mu_B(x)$$

协 T 算子 \oplus 是满足下列条件的一个两变量函数 $S(\cdot, \cdot)$ 。

- 单调：如果 $a \leq c$ 且 $b \leq d$ ，则 $S(a, b) \leq S(c, d)$ 。
- 右界： $S(1, 1) = 1$ ， $S(a, 0) = S(0, a) = a$ 。
- 交换律： $S(a, b) = S(b, a)$ 。
- 结合律： $S(a, S(b, c)) = S(S(b, a), c)$ 。

常用的 T 算子和协 T 算子定义如下。

- T 算子

$$\mu_A(x) \otimes \mu_B(x) = \begin{cases} \min(\mu_A(x), \mu_B(x)), & (\text{模糊并}) \\ \mu_A(x) \cdot \mu_B(x), & (\text{代数积}) \\ \max(0, (\mu_A(x) + \mu_B(x) - 1)), & (\text{有界积}) \\ \left. \begin{array}{l} \mu_A(x), \quad \mu_B(x) = 1 \\ \mu_B(x), \quad \mu_A(x) = 1 \end{array} \right\} & (\text{直积}) \\ 0, \quad \mu_A(x) < 0, \mu_B(x) < 0 \end{cases}$$

- 协 T 算子

$$\mu_A(x) \oplus \mu_B(x) = \begin{cases} \max(\mu_A(x), \mu_B(x)), & (\text{模糊并}) \\ \mu_A(x) + \mu_B(x), & (\text{代数和}) \\ \min(1, (\mu_A(x) + \mu_B(x))), & (\text{有界和}) \\ \left. \begin{array}{l} \mu_A(x), \quad \mu_B(x) = 1 \\ \mu_B(x), \quad \mu_A(x) = 1 \end{array} \right\} & (\text{直和}) \\ 0, \quad \mu_A(x) > 0, \mu_B(x) > 0 \end{cases}$$

4. 模糊规则

在模糊推理系统工程中，模糊规则以模糊语言的形式描述人类的经验和知识，规则是否正确反映人类专家的经验和知识更新，是否能反映对象的特性，直接决定了模糊推理系统的性能，通常模糊规则的形式是 if...then，前提由对模糊语言变量的语言描述构成，如“温度较高”，“压力较低”等，结论由对输出模糊语言变量表示成输入量的精确的组合，模糊规则的这种形式化表示符合人们对许多知识的描述和记忆习惯。

模糊规则的建立是构造模糊推理系统的关键,其建立方法主要有如下3种。

① 总结操作人员、专家的经验 and 知识。操作人员在长期从事仪器设备的操作中,积累了大量的经验,这些经验都具有模糊性的特点。总结这些经验对构造模糊规则有重要的指导意义。某个领域的专家则对该领域的各种过程机理有较深刻的认识,对过程的运行特性能够通过理论分析给出定性的结论以帮助建立模糊规则。

② 基于过程的模糊模型。被控过程的动态特性可以用模糊模型来描述,称为过程的模糊模型。基于过程的模糊模型可以产生一组模糊控制规则来使被控过程到达期望的性能。这种方法存在的困难就是难以获得能够充分反映被控过程特性的模糊模型及参数。

③ 基于学习的方法。当被控过程存在时变的特性或难以直接构造模糊控制器时,可以通过设计具有自学习能力的模糊控制器来自动获得模糊规则。Procyk 和 Mamdani 首先提出了自组织、自学习能力的模糊控制器的一种分级结构,包括两级控制规则:第一级直接用于控制对象;第二级热气测量数据和评价准则在线修改第一级模糊规则。

上面介绍了建立模糊规则的3种主要方法,其中第一种方法是最基本的,也是应用最广泛的方法。在实际应用中,初步建立的模糊规则往往不能达到良好的效果,必须不断加以修正和试凑。在模糊规则的建立、修正和试凑过程中,应尽量保证模糊规则的完备性和相容性。所谓模糊规则的完备性,是指对于控制过程的任一状态,模糊规则都能产生有关的控制作用。模糊规则的相容性则反映在输出模糊集合是否是多峰的,如果存在多峰的现象,则说明模糊规则中有相互矛盾的情况存在。

最简单的 if...then 规则的形式是:“如果 x 是 A , 则 y 是 B ”。复合型的 if...then 规则的形式很多,分别如下所示。

- if m 是 A 且 x 是 B then y 是 C , 否则 z 是 D 。
- if m 是 A 且 x 是 B 且 y 是 C , then z 是 D 。
- if m 是 A 或 x 是 B then y 是 C , 或 z 是 D 。
- if m 是 A 且 x 是 B then y 是 C , 且 z 是 D 。

其中, A 、 B 、 C 、 D 分别是论域 m 、 x 、 y 、 z 中模糊集的真值, if 部分是前提或前件, then 部分是结论或后件。解释 if...then 规则包括以下3个过程。

- ① 输入模糊化。确定出 if...then 规则前提中每个命题或断言为真的程度(即隶属度)。
- ② 应用模糊算子。利用模糊算子可以确定整个前提为真的程度(即整个前提的隶属度)。
- ③ 应用蕴涵算子。由前提的隶属度和蕴涵算子,可以确定结论为真的程度(即结论的隶属度)。

5. 模糊推理

模糊推理是采用模糊逻辑由给定的输入到输出的映射过程。模糊推理包括5个过程。

- ① 输入变量模糊化,即把确定的输入转化为由隶属度描述的模糊集。
- ② 在模糊规则的前件中应用模糊算子(与、或、非)。
- ③ 根据模糊蕴涵运算由前提推断结论。
- ④ 合成每一个规则的结论部分,得出总结的结论。
- ⑤ 反模糊化,即把输出的模糊量转化为确定的输出。

6. 模糊控制

在自动控制理论(包括现代控制理论)中,控制器的分析与综合依赖于精确的数学模型。由于被控对象过程的非线性、参数间的强烈耦合、较大的随机干扰、过程机理错综复杂,以及

现场测量仪表条件的不足,或者测试仪表无法进入被测区,以致不可能建立起被控对象的数学模型,对于那些不能直接获得数学模型描述的系统,传统的控制方法难以取得令人满意的控制效果,然而这类被控对象在手工控制下却能够正常运行,达到一定的预期效果。

在手工控制中,操作人员在长期观察、实践中积累许多经验,这些经验常用定性的、不精确的语言规则等形式加以描述,如“若炉温偏高,则燃料适当减少”。系统在运行过程中,人们将观察到的过程输出与设定值比较,得到过程输出偏离设定值程度的模糊语义描述或过程输出偏离设定值变化快慢的模糊语义描述,经逻辑推理得出控制量的模糊量:“适量减少燃料”,再经反模糊化且转化为一精确的控制量,实现整个控制过程。以模糊集和模糊推理为基础,对上述手工操作过程进行建模,得到模糊控制器。

模糊系统除用于自动控制外,还用于模糊聚类、建模、信号处理、计算机视觉、专家系统、决策分析、图像处理等许多领域,其理论基础主要是模糊推理,但是具体的问题所采用的隶属度函数形式和模糊算子的形式不同。

6.1.2 模糊推理系统与MATLAB应用

1. 模糊推理系统结构

最常见的模糊推理系统分为 3 类。

① 纯模糊逻辑系统,它的输入与输出均为模糊集合,模糊推理在模糊推理系统中起着核心作用,它将输入模糊集合按照模糊规则映射为输出模糊集合。

② Sugeno 型模糊逻辑系统,它的输出量在没有模糊消除的情况下仍然是精确值,其输出量由输入值的线性组合来表示。

③ Mamdani 型模糊系统,是具有模糊产生器和模糊消除器的模糊逻辑系统,是最广泛实用的模糊系统,通常简称为模糊逻辑系统,其示意图如图 6-1 所示。

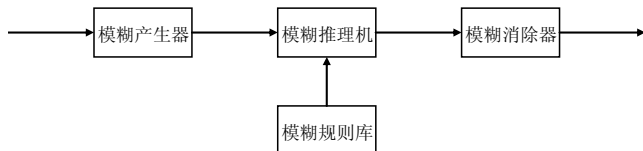


图 6-1 Mamdani 型模糊系统示意图

2. Mamdani型模糊逻辑系统构建

Mamdani 型模糊逻辑系统是典型的模糊逻辑系统,MATLAB 模糊逻辑工具箱中的模糊推理系统有 5 个过程:输入变量的模糊化、模糊关系运算、模糊合成运算、不同规则结果的综合、去模糊化。

关于 Mamdani 型模糊逻辑系统的构建在此不展开介绍,在后面介绍中会涉及相关内容,有需要了解这方面知识的读者请参看帮助文档。

6.1.3 模糊推理系统的MATLAB模糊工具箱的图形界面实现法

模糊推理系统可通过 MATLAB 模糊工具箱的图形界面工具来实现,方法简单并且直观。也可利用 MATLAB 提供的命令行方式的模糊逻辑函数编辑实现,这种方法有利于实现比较复杂的模糊推理系统。MATLAB 模糊工具箱的图形化工具与命令行函数是统一的,其格式都是相同的。

1. 图形界面工具箱简介

MATLAB 模糊工具箱提供的图形化工具包括如下 5 类。

- Fuzzy: 模糊推理系统编辑器。
- Mfedit: 隶属度函数编辑器。
- Ruleedit: 模糊规则编辑器。
- RuleView: 模糊规则观察器。
- Surfview: 模糊推理输入/输出曲面视图。

这 5 个图形工具操作规程简单, 相互动态联系, 可以同时用来快速构建用户设计的模糊系统。

Fuzzy 用来处理系统的最顶层的构建问题, 例如输入/输出变量的数目、变量名等。通常 MATLAB 并不限制输入的数目, 但是对于复杂的大系统, 输入可能会受到计算机内存的限制。在输入的数目太多或者模糊规则数目太多的情况下, 使用图形化工具就会比较困难, 就需要通过编写相应的程序来完成。Mfedit 用来可视化定义各个变量的隶属度函数。Ruleedit 用来编辑决定系统输出的模糊规则。Ruleview 和 Surfview 用来查看规则和模糊推理系统的输入/输出关系曲面, 它们用来计算、显示、模拟、分析和诊断系统, 具有只读属性, 并不对系统进行修改。

2. Fuzzy (模糊推理系统编辑器)

基本模糊推理系统编辑器提供了利用 GUI (图形界面) 对模糊系统的高层属性的编辑和修改功能, 包括输入/输出语言变量的个数和去模糊化方法等。在基本模糊编辑器中可以通过菜单选择激活其他几个图形界面编辑器, 如 Ruleedit (模糊规则编辑器)、Mfedit (隶属度函数编辑器) 等。在 MATLAB 命令窗口执行 Fuzzy 命令即可激活基本模糊推理系统编辑器, 其图形界面如图 6-2 所示。

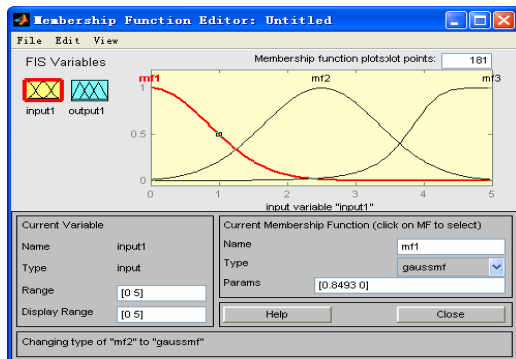


图 6-2 模糊推理系统编辑器

在窗口上半部以图形框的形式列出了模糊推理系统的基本组成部分, 即输入模糊变量、模糊规则和输出模糊变量。通过双击上述图形框, 能够激活隶属度函数编辑器和模糊规则编辑器等相应的编辑窗口。在窗口的下半部分的左侧列出了模糊推理系统的名称、类型和一些基本属性, 包括“与”运算方法、“或”运算方法、蕴涵运算、模糊规则的综合运算以及去模糊化方法等, 窗口下半部分的右侧, 列出了当前选定的模糊语言变量的名称及其论域范围。在实际中, 根据需要来调整选取以输出理想结果。

在 Fuzzy 的菜单部分主要提供了如下功能。

① File (文件) 菜单的主要功能如下。

- New Msmdani FIS: 新建 Msmdani 型模糊推理系统。
- New Sugeno FIS: 新建 Sugeno 型模糊推理系统。

- Open FIS From Disk: 从磁盘打开一个 FIS 型模糊推理系统文件。
- Save to disk: 将当前的模糊推理系统保存到磁盘中。
- Save As to disk: 将当前的模糊推理系统另存为一个文件。
- Open FIS From Workspace: 从工作空间加载一个 FIS 型模糊推理系统。
- Save to Workspace: 保存到工作空间。
- Save to Workspace as: 另存到工作空间的某一模糊推理系统。
- Print: 打印模糊推理系统的信息。
- Close Window: 关闭窗口。

② Edit (编辑) 菜单的主要功能如下。

- Add input: 添加输入语言变量。
- Add output: 添加输出语言变量。
- Remove variable: 删除语言变量。

③ View (视图) 菜单主要功能如下。

- Edit FIS Properties: 修改模糊推理系统的特性。
- Edit membership functions: 打开隶属度函数编辑器。
- Edit Rules: 打开模糊规则编辑器。
- View Rules: 打开模糊规则浏览器。
- View Surface: 打开模糊系统输入/输出特性浏览器。

MATLAB 模糊工具箱中已经附带了很多示例模型。这里以 MATLAB 工具箱中的典型示例说明模糊系统的构建过程。

【例 6-1】小费问题。

一般情况下, 顾客在饭店里所给的小费与服务以及食物质量有关。简化问题模型的输入为服务质量和食物质量, 输入的模糊空间范围, 即输入论域是顾客给出各项的分数分别是食物为 0~10 分, 服务为 0~10 分。所给的评语分别是, 食物为“差, 好”; 服务为“差, 好, 很好”; 输出的模糊空间范围, 即输出论域是小费为 (总价格的) 5%~20%, 其评语为“少, 中, 高”。系统有如下 3 条模糊规则。

- ① 如果服务差, 食物差, 则小费低。
- ② 如果服务好, 食物差, 则小费中。
- ③ 如果服务好和食物非常可口, 则小费高。

小费问题, 在 MATLAB 中已经提供了几种现成的模糊逻辑推理系统方案, 分别存为 `custtip.fis`, `tipper.fis`, `tipper1.fis`, `tippersg.fig`。下面以系统 `tipper.fis` 为例讲解图形编辑工具的使用。

(1) 直接读取磁盘中的文件的方法

在 MATLAB 中输入命令:

```
>> fuzzy tipper (或 fuzzy tipper.fis)
```

打开模糊系统 `tipper.fis` 的编辑窗口, 如图 6-3 所示。

图形化模糊系统工程工具 `fuzzy` 函数已经将在磁盘上的小费问题的模糊推理系统 `tipper.fis` 读入了内存, 这时就可以用前面提到的 5 类图形化工具来对这个模糊系统进行计算、模拟、实现、修改等操作。

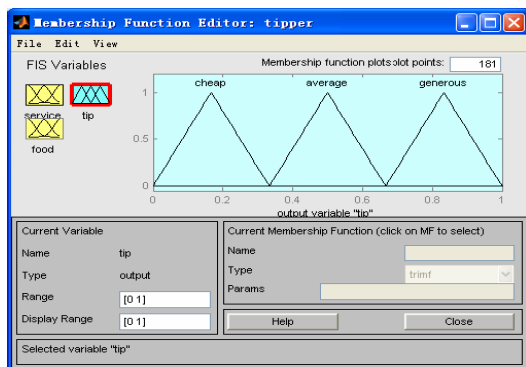


图 6-3 隶属度函数编辑窗口

(2) 自建fis文件的方法

- ① 在 MATLAB 中输入命令 `fuzzy f` 进入模糊系统, 得到基本模糊推理系统编辑器图形界面。
- ② 单击【Edit】菜单下的【Add input】选项, 使系统变成两个输入、一个输出, 此时的文件名为 `untitled`。
- ③ 单击系统的 `input1` 输入文本框, 在 GUI 的右下角的空白处修改输入名称为 `service`, 同理将 `input2` 输入文本框修改为 `food`, 通常 FIS 编辑器默认为 Mamdani 型系统。
- ④ 单击系统的 `output1` (输出文本框), 在 GUI 的右下角的空白处修改输出框的名称作 `tip`。
- ⑤ 单击【File】菜单下的【Save to Workspace as ...】, 并输入变量名为 `tip`, 这时在 MATLAB 的工空间中得到一个 FIS 结构, 名为 `tipper`。

```
>> tipper
tipper =
      name: 'tipper'
      type: 'mamdani'
    andMethod: 'min'
    orMethod: 'max'
  defuzzMethod: 'centroid'
    impMethod: 'min'
    aggMethod: 'max'
      input: [1x2 struct]
      output: [1x1 struct]
       rule: []
```

这样就建立了初步的模糊推理系统 GUI, 与图 6-4 的不同在于没有 Rule 输入, 其内容为空白。

在操作中, 选择【Save to Workspace as ...】将其存到工作空间内, 而不是存入磁盘。如果选择存入磁盘的话, 应当注意避免和系统中已有的文件重名。编辑窗口先后进行模糊系统中的相关隶属度函数及模糊规则等内容的编辑。

模糊隶属度函数的编辑有 3 种方式打开编辑窗口。

- ① 单击【View】菜单下的【Edit membership function】项。
- ② 用双击窗口中需要编辑的变量图标, `tip`。
- ③ 直接在命令行中输入命令 `mfedit`。

3. Mfedit (隶属度函数编辑器)

在命令窗口输入 `mfedit`，或在基本模糊推理系统编辑器中选择编辑隶属度函数的菜单，都可以激活隶属度函数编辑器。该编辑器提供了对输入/输出语言变量各语言值的隶属度函数类型、参数进行编辑与个性的图形界面工具。在模糊推理系统编辑器菜单中单击【View】菜单下的【Edit membership function】选项，如图 6-4 所示。

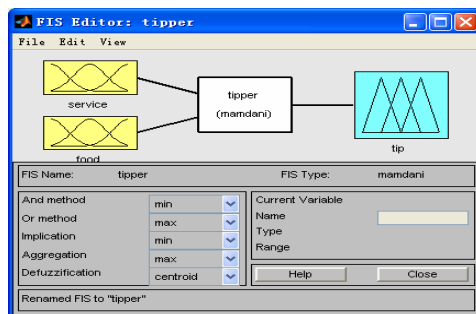


图 6-4 小费问题模糊系统编辑窗口

窗口上半部分为隶属度函数的图形显示；下半部分为隶属度函数的参数设定界面，包括语言变量的名称、论域和隶属度函数的名称、类型和参数。在菜单部分，文件菜单和视图菜单的功能与模糊推理系统编辑器的文件功能类似。Edit 菜单的功能包括添加定制的隶属度函数以及删除隶属度函数等。Edit 菜单的功能如下。

Add Mfs...
Add Custom MF...
Remove Current MF
Remove All MF

按下面的参数来修改文件：

```
service
Name='service'
Range=[0 10]
NumMFs=3
Name          type          params
MFs='poor'    'gaussmf',    [1.5 0]
MFs='good'    'gaussmf',    [1.5 5]
MFs='excellent' 'gaussmf',    [1.5 10]
food
Name='food'
Range=[0 10]
NumMFs=2
Name          type          params
MF1='rancid'  'trapmf',     [0 0 1 3]
MF2='delicious' 'trapmf',     [7 9 10 10]
Tip
Name='tip'
Range=[0 30]
NumMFs=3
Name          type          params
```

MF1='cheap'	'trimf',	[0 5 10]
MF2='average'	'trimf',	[10 15 20]
MF3='generous'	'trimf',	[20 25 30]

将改动后的文件保存到工作空间，完成上述操作，如图 6-5 所示。

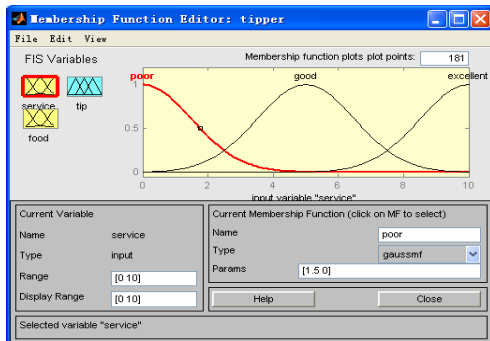


图 6-5 经过编辑后的隶属度函数

4. Ruleedit（模糊规则编辑器）

在 MATLAB 命令窗口输入 ruleedit 命令，或在基本模糊推理系统编辑器中选择编辑模糊规则菜单，均可激活模糊规则编辑器。在模糊规则编辑器中，提供了添加、修改和删除模糊规则的图形界面。在模糊推理系统编辑器菜单中，单击【View】菜单下的【Edit Rules...】选项或双击 FIS Edit 窗口中间白色的模糊规则图标，打开模糊规则编辑器，如图 6-6 所示。

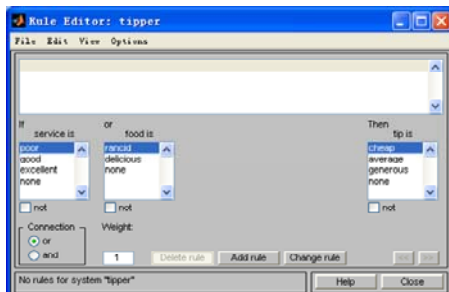


图 6-6 模糊规则编辑器

在模糊规则编辑器中提供了一个文本框，用于规则的输入和修改。模糊规则的显示方式包括 3 种：Verbose（语言型）、Symbolic（符号型）以及 Indexed（索引型）。

模糊规则编辑器的菜单功能与前两个基本编辑器类似，在其视图菜单中能够激活其他的编辑器或窗口。

窗口下部还有 3 个按钮，分别为 Delete rule（删除规则）、Add rule（增大规则）及 Change rule（改变规则）。

在这个窗口下，编辑模糊规则是十分方便的，系统已经自动地将在 FIS dit 中定义的变量显示在界面的左下部。在窗口上选择相应的输入变量（以及是否加否定词 not），然后选择不同变量之间的连接关系（or 或者 and）以及输入权重（默认为 1）。然后，单击 Add rule 按钮，输入的规则就在编辑器上面的显示区域中出现了。

例如，在“service”列表中选择“poor”项，在“food”列表中选择“rancid”项，在“connection”项中选择“or”单选按钮，单击“Add rule”按钮，出现如下结果。

1. If (service is poor) or (food is rancid) then (tip is cheap) (1)

括号中的数字是该规则的权重值。依次输入如下内容。

2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)

输入完成之后，文本框内出现刚加入的模糊推理规则，如图 6-7 所示。

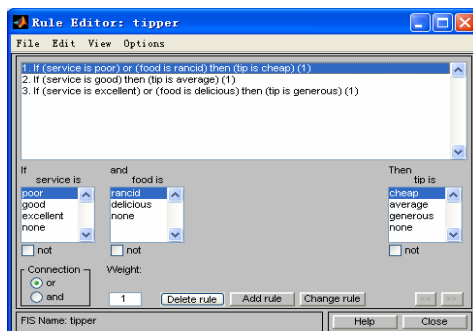


图 6-7 输入规则后的模糊规则编辑器

也可以从菜单“Options”项中选择相应的显示方式。

将显示方式设为 symbolic，显示如下：

1. (service==poor) | (food==rancid) => (tip==cheap) (1)
2. (service==good) => (tip==average) (1)
3. (service==excellent) | (food==delicious) => (tip==generous) (1)

如设为 indexed，显示如下。

- 1 1, 1 (1) : 2
- 2 0, 2 (1) : 1
- 3 2, 3 (1) : 2

这 3 种显示方式虽然不同，但是规则内部的实际含义仍然是相同的。

5. Ruleview（模糊规则观察器）

在 MATLAB 命令窗口输入 ruleview 命令，或在上述 3 种编辑器中选择相应菜单，都可以激活模糊规则观察器。在模糊规则观察器中，以图形描述了模糊推理系统的推理过程，如图 6-8 所示。在“Rule Viewer: tipper”窗口中，上部是各输入/输出在输入选取确定值后的各评价语隶属度情况和按输入的规则推理运算后的各对应输出隶属度情况。窗口的下部有输入值调整窗口，可以在该窗口内改变输入的相应的参数来观察模糊逻辑推理系统的输出的情况。

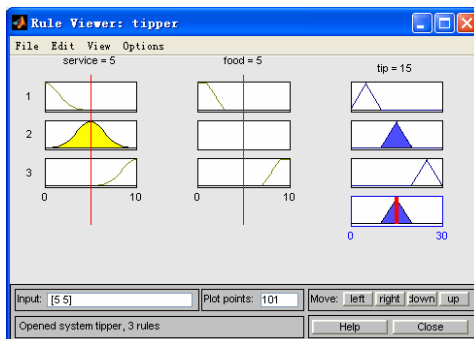


图 6-8 模糊规则演示窗口

6. Surfview (模糊推理输入/输出曲面视图)

在 MATLAB 命令窗口输入 surfview 命令, 或在各个编辑器窗口选择相应菜单, 打开模糊推理输入/输出曲面视图窗口。该窗口以图形的形式显示了模糊推理系统的输入/输出特性曲面。在命令窗口中输入命令 surfview tipper, 打开“Surface Viewer: tipper”窗口, 如图 6-9 所示。

以上是利用 MATLAB 模糊工具箱中的图形界面工具完成模糊推理系统的构建过程, 整个过程简单方便并且直观。

当然, MATLAB 模糊工具箱中也有其他的方法可以构建模糊推理系统, 也有一些特殊功能的函数, 如自定义函数, 这里不做介绍。

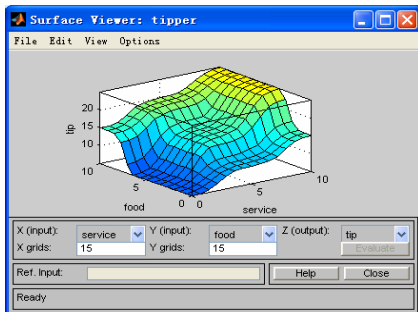


图 6-9 曲面观察器

6.2 MATLAB模糊逻辑工具箱命令函数及示例

1. 隶属度函数

(1) dsigmf函数

功能: 由两个 S 型隶属度函数的差构成的隶属度函数。

格式: $y = \text{dsigmf}(x, [a1 \ c1 \ a2 \ c2])$

说明: 这里使用的 S 型隶属度函数取决于 a 和 b 两个参数, 其公式如下。

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}}$$

隶属度函数 dsigmf 有 4 个参数, a1, c1, a2 和 c2, 它是隶属度函数的差。

$$f1(x, a1, c1) - f2(x, a2, c2)$$

$f1$ 和 $f2$ 可参见 sigmf 函数。

(2) psigmf函数

功能: 由两个 S 型隶属度函数的积构成的隶属度函数。

格式: $y = \text{psigmf}(x1, [a1 \ c1 \ a2 \ c2])$

说明: S 型函数由参数 a 和 c 确定, 其公式如下。

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}}$$

psigmf 函数只是两个这种 S 型函数的乘积, 公式如下。

$$f1(x, a1, c1) \times f2(x, a2, c2)$$

其参数由[a1 c1 a2 c2]指定。

(3) gaussmf 函数

功能：高斯型隶属度函数。

格式：y=gaussmf(x,[sig c])

说明：对称的高斯函数取决于两个参数， σ （用 sig 表示）和 c，其公式如下。

$$f(x, \sigma, c) = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

gaussmf 函数的参数以向量[sig c]形式给出。

【例 6-2】gaussmf 函数示例。

```
x=0:0.1:10;
y=gaussmf(x,[2 6]);
plot(x,y);
xlabel('gaussmf,p=[2 6]')
```

运行程序，得到高斯型隶属度函数如图 6-10 所示。

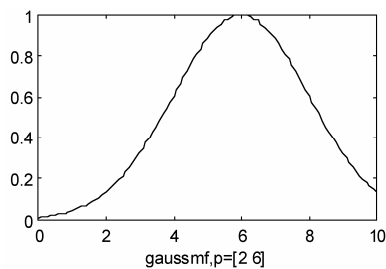


图 6-10 高斯型隶属度函数

(4) gauss2mf 函数

功能：联合高斯型隶属度函数。

格式：y=gauss2mf(x,[sig1 c1 sig2 c2])

说明：高斯函数取决于两个参数， σ （用 sig 表示）和 c，其公式如下。

$$f(x, \sigma, c) = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

函数 gauss2mf 是两个高斯函数的联合，第一个函数由 sig1 和 c1 指定，第二个函数由 sig2 和 c2 指定，分别用于指定左边和右边的形状。只要 $c1 < c2$ ，则 gauss2mf 的最大值达到 1；否则 gauss2mf 的最大值小于 1。

(5) primf 函数

功能：II 型隶属度函数。

格式：y=primf(x,[a b c d])

说明：这种基于样条的曲线因其形状而得名。primf 函数可在指定向量 x 处计算隶属度函数值，参数 a 和 d 用于确定曲线的“脚”，参数 b 和 c 用于确定曲线的“肩膀”。

(6) gbellmf 函数

功能：广义钟型隶属度函数。

格式：y=gbellmf(x,params)

说明：广义钟型函数有 3 个参数 a, b, c，其公式如下。

$$f(x, a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2\sigma}}$$

其中，参数 b 通常为正值，参数 c 用于确定曲线的中心。在 `gbellmf` 函数的第二个参数 `params` 中应输入向量，其内容分别为 a 、 b 和 c 。

(7) smf函数

功能：S 状隶属度函数。

格式：`y=smf(x,[a b])`

说明：这是基于样条曲线的函数，因其形状而得名，参数 a 和 b 用于确定曲线的形状。

(8) trapmf函数

功能：梯形隶属度函数。

格式：`y=trapmf(x,[a b c d])`

说明：梯形曲线可由 4 个参数 a 、 b 、 c 、 d 确定，其公式如下。

$$f(x, a, b, c, d) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{c-x}{c-b}, & c \leq x \leq d \\ 0, & x \geq d \end{cases}$$

或者更紧凑地表示形式如下。

$$f(x, a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}, 0\right), 0\right)$$

其中，参数 a 和 d 确定梯形的“脚”，而参数 b 和 c 确定梯形的“肩膀”。

【例 6-3】`trapmf` 函数示例。

```
x=0:0.1:10;
y=trapmf(x,[1 5 7 9]);
plot(x,y);
xlabel('trapmf,p=[1 5 7 9]');
```

运行程序，可得到梯形隶属度函数曲线，如图 6-11 所示。

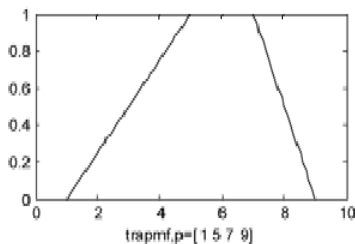


图 6-11 梯形隶属度函数

(9) sigmf函数

功能: S 型隶属度函数。

格式: $y = \text{sigmf}(x, [a \ c])$

说明: S 型函数 $\text{sigmf}(x, [a \ c])$ 由参数 a 和 c 决定, 其公式如下。

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}}$$

根据参数 a 的正负, 可确定 S 型隶属度函数的开口朝左或朝右, 这正好可用来表示“正很大”或“负很大”的概念, 更一般的隶属度函数可由两个不同的 S 型隶属度函数的积或差来构成。

(10) zmf函数

功能: Z 型隶属度函数。

格式: $y = \text{zmf}(x, [a \ b])$

说明: 这是基于样条曲线的函数, 因其呈现 Z 形状而得名。参数 a 和 b 确定曲线的形状。

【例 6-4】zmf 函数示例。

```
x=0:0.1:10;
y=zmf(x,[3 7]);
plot(x,y);
xlabel('zmf,p=[3 7]');
```

运行程序, 得到 Z 型隶属度函数曲线, 如图 6-12 所示。

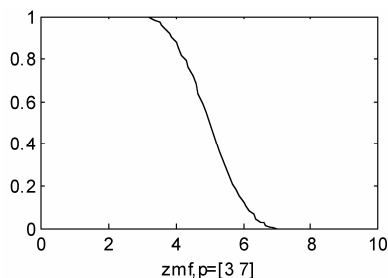


图 6-12 Z 型隶属度函数

(11) trimf函数

功能: 三角形隶属度函数。

格式: $y = \text{trimf}(x, \text{params})$ 或 $y = \text{trimf}(x, [a \ b \ c])$

说明: 三角形曲线由 3 个参数 a , b , c 确定, 其公式如下。

$$f(x, a, b, c, d) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases}$$

或者更紧凑地表示如下。

$$f(x,a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{c-x}{c-b}\right), 0\right)$$

参数 a 和 c 确定三角形的“脚”，而参数 b 确定三角形的“峰”。

【例 6-5】trimf 函数示例。

```
x=0:0.1:10;
y=trimf(x,[2 5 8]);
plot(x,y);
xlabel('trimf,p=[2 5 8]');
```

运行程序，得到三角形隶属度函数曲线，如图 6-13 所示。

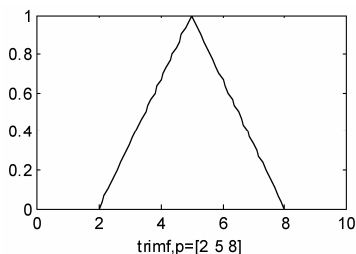


图 6-13 三角形隶属度函数

2. FIS数据结构管理

(1) addvar函数

功能：在 FIS 中添加变量。

格式：a=addvar(a,'varType','varName',varBounds)

说明：addvar 函数的 4 个输入变量如下。

a：工作空间中的 FIS 的变量名。

varType：添加隶属度函数的变量类型（即 input 或 output）。

varName：添加的变量名。

varBounds：变量的取值范围。

添加的变量按其添加的顺序进行编号，这样添加到系统的第一个变量总是称为系统的输入变量 1，输入与输出变量单独编号。

【例 6-6】addvar 函数示例。

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
getfis(a,'input',1)
    Name =      service
    NumMFs =      0
    MFLabels =
    Range =      [0 10]
```

输出结果为：

```
ans =
    Name: 'service'
```

NumMFs: 0
range: [0 10]

(2) addmf 函数

功能：隶属度函数添加到 FIS（模糊推理系统）。

格式：a=addmf(a,'varType',varIndex,'mfName','mfType',mfParams)

说明：隶属度函数只能添加到 MATLAB 工作空间中已建立的 FIS 结构中。按隶属度函数添加的顺序将其编号，这样给变量添加的第一个隶属度函数称为该变量的 1 号隶属度函数。

addmf 函数的 6 个输入变量分别如下。

a：工作空间中的 FIS 的变量名。

varType：要添加隶属度函数的变量类型（即 input 或 output）。

varIndex：要添加隶属度函数的变量编号。

mfName：新隶属度函数名。

mfType：新隶属度函数的类型。

mfParams：指定隶属度函数的参数向量。

【例 6-7】addmf 函数示例。

```
a=newfis('tipper'); %建立新的 FIS 系统
a=addvar(a,'input','service',[0 10]); %给 FIS 添加新的输入变量 service
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
plotmf(a,'input',1);
```

运行程序，效果如图 6-14 所示。

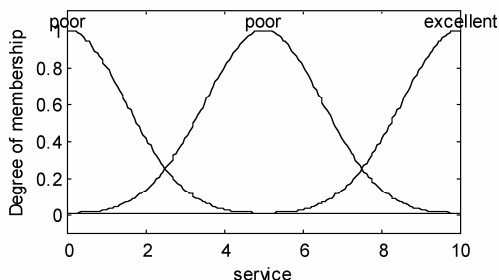


图 6-14 隶属度函数曲线

(3) addrule 函数

功能：在 FIS 中添加规则。

格式：a=addrule(a, ruleList)

说明：addrule 函数有两个变量，第一个变量 a 为 FIS 的变量名，第二个变量 ruleList 表示规则的矩阵。规则列表矩阵的格式有严格的要求，当模糊系统有 m 个输入、n 个输出时，规则列表矩阵有 m+n+2 列，前 m 列表示系统的输入，每列的数值表示输入变量隶属度函数的编号；接着 n 列表示系统的输出，第 2 列的数值表示输出变量隶属度函数的编号；第 m+n+1 列的内容为该条规则的权值（0~1）；第 m+n+2 列的值决定模糊操作符的类型；当模糊操作为 and 时，为 1 或当模糊操作为 or 时，为 0。

【例 6-8】addrule 函数示例。

```
ruleList=[1 1 1 1 1
          1 2 2 1 1];
a=addrule(a,ruleList);
```

如果系统 a 有两个输入和一个输出，则上述定义的第一条规则为：If X is x1 and Y is y1 then Z is z1。

(4) defuzz 函数

功能：反模糊化的隶属度函数。

格式：out=defuzz(x, mf, type)

说明：defuzz(x, mf, type)可得到输入为 x 时的隶属度函数 mf 的反模糊化值，其反模糊化的策略由 type 指定。变量 type 如下。

centroid：区域重心法。

bisector：区域等分法。

mon：极大平均法。

som：极大最小法。

lom：极大最大法。

如果 type 不取上述各种方法，则默认为用户自定义的方法，x 和 mf 通过函数可产生反模糊化的结果。

【例 6-9】defuzz 函数示例。

```
>> x=-10:0.1:10;
mf=trapmf(x,[-10 -8 -4 7]);
xx=defuzz(x,mf,'centroid')
xx =
    -3.2857
```

(5) evalmf 函数

功能：普通隶属度函数的计算。

格式：y=evalmf(x, mfParams, 'mfType')

说明：evalmf 函数可计算任意的隶属度函数，其中，x 为要计算的隶属度函数取值，mfType 为工具箱中存在的一种隶属度函数，mfParams 为该函数的合适参数。如果建立了自己的隶属度函数，evalmf 函数也能很好地工作，这是因为 evalmf 只计算隶属度函数，并不对其名字进行识别。

【例 6-10】evalmf 函数示例。

代码如下：

```
x=0:0.1:10;
mparams=[2 5 8];
mtype='gbellmf';
y=evalmf(x,mparams,mtype);
plot(x,y);
xlabel('gbellmf,p=[2 5 8]');
```

运行程序，效果如图 6-15 所示。

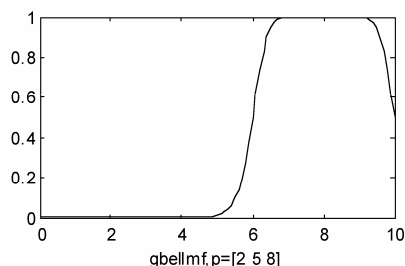


图 6-15 普通隶属度函数计算结果曲线

(6) evalfis 函数

功能：完成模糊推理计算。

格式：output=evalfis(input,fismat);

output=evalfis(input,fismat,numPts);

[output,IRR,ORR,ARR]=evalfis(input,fismat);

[output,IRR,ORR,ARR]=evalfis(input,fismat,numPts)

说明：evalfis 函数具有如下参量。

input：指定输入的数值或矩阵。如果输入为 $m \times n$ 的矩阵时 (n 为输入变量数)，则 evalfis 将输入的第一行看成输入向量，并在输出变量 output 中产生 $m \times 1$ 矩阵，其中每一行为输出向量，1 为输出变量数。

fismat：要计算的 FIS 结构。

numPts：计算输入和输出隶属度函数时采用的取样点数，如果默认，则采用默认值 101。

output：evalfis 函数的输出变量为 $m \times 1$ 的矩阵，其中 m 表示输入变量数，1 表示输出变量数。

IRR：输入值通过隶属度函数后的结果，这是 $\text{numRules} \times n$ 的矩阵，其中 numRules 为规则数， n 为输入变量数。

ORR：输出值通过隶属度函数后的结果，这是 $\text{numPts} \times \text{numRules} \times 1$ ，其中 numRules 为规则数，1 为输出变量数。这个矩阵的前 numRules 列对应于第一个输出，接下来的 numRules 列对应于第二个输出。

ARR：沿着每个输出的取值范围以 numPts 取样得到的 $\text{numPts} \times 1$ 矩阵。

只有当输入变量为行向量（只使用一组输入值）时，evalfis 才可得到输出变量的值，只带一个输出向量 output。

【例 6-11】evalfis 函数示例。

```
fismat=readfis('tipper');
output=evalfis([2 3;5 9],fismat)
```

输出结果为：

```
output =
    7.7885
   19.9392
```

(7) gensurf 函数

功能：产生 FIS 输出曲面。

格式：gensurf(fis);

```
gensurf(fis, input, output);
gensurf(fis, input, output, grids);
gensurf(fis, input, output, grids, reinput);
[x, y, z]=gensurf(...)
```

说明: gensurf(fis)函数针对给定 FIS 的前两个输入和第一个输出绘制曲面。

gensurf(fis, input, output)可在绘制输出曲面时采用指定的输入（由 input 指定）和输出（由标量 output 指定）。

gensurf(fis, input, output, grids)还可指定 X 和 Y 方向的栅格数, 如果 grids 为二元向量, 则可独立指定 X 和 Y 的栅格数。

gensurf(fis, input, output, grids, reinput)可有两个以上的输出, 其中 reinput 指定系统不变的输入。

[x, y, z]=gensurf(...)得到输出曲面的坐标值, 并抑制自动显示。

【例 6-12】gensurf 函数示例。

```
a=readfis('tipper');
gensurf(a);
```

运行程序, 效果如图 6-16 所示。

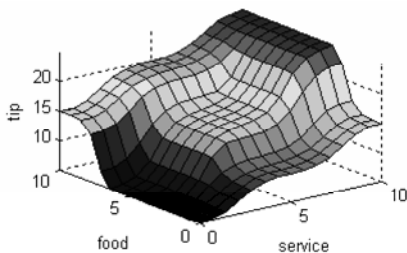


图 6-16 FIS 输出曲面

(8) getfis函数

功能: 获取模糊系统的特性。

格式: getfis(a);

```
getfis(a,'fisprop');
```

```
getfis(a,'VarType',varIndex,'varProp');
```

```
getfis(a,'VarType',varIndex,'mf',mfIndex);
```

```
getfis(a,'VarType',varIndex,'mf',mfIndex,'mfProp');
```

说明: 这是 FIS 结构的基本访问函数, 利用这一函数可获得 FIS 的每个部分。getfis 函数的输入变量如下。

a: FIS 结构的变量名。

VarType: 变量类型的字符串, 可取 input 或 output。

varIndex: 变量的整数, 例如, 1 表示输入 1 或输出 1。

mf: 要搜索的隶属度函数信息的字符串。

mfIndex: 要搜索信息的隶属度函数的序号。

【例 6-13】getfis 函数示例。

① 单输入变量。

```
>>a=readfis('tipper');
>>getfis(a)
    Name      = tipper
    Type      = mamdani
    NumInputs = 2
    InLabels   =
        service
        food
    NumOutputs = 1
    OutLabels  =
        tip
    NumRules = 3
    AndMethod = min
    OrMethod  = max
    ImpMethod = min
    AggMethod = max
    DefuzzMethod = centroid
```

② 双输入变量。

```
>> getfis(a,'type')
ans =  mamdani
```

③ 3 个输入变量。

```
>>getfis(a,'input',1)
    Name =      service
    NumMFs =      3
    MFLabels =
        poor
        good
        excellent
    Range =      [0 10]
```

④ 4 个输入变量。

```
>> getfis(a,'input',1,'name')
ans =  service
```

⑤ 5 个输入变量。

```
>> getfis(a,'input',1,'mf',2)
    Name = good
    Type = gaussmf
    Params = [1.5 5]
```

⑥ 6 个输入变量。

```
>> getfis(a,'input',1,'mf',2,'name')
ans =  good
```

(9) newfis函数

功能：建立新的 FIS。

格式：a=newfis(fisName,fisType,andMethod,orMethod,impMethod,aggMethod)

说明：这一函数可建立新的 FIS 结构，newfis 函数最多可有 7 个输入变量，其输出变量为 FIS 结构。7 个输入变量分别如下。

fisName: FIS 结构名，其后缀默认为.fis。

fisType: FIS 类型。

andMethod、orMethod、impMethod、aggMethod 和 defuzzMethod 分别指定 and、or、蕴涵、聚集和反模糊化运算方法。

【例 6-14】newfis 函数示例。

```
>> a=newfis('newsys');
getfis(a)
    Name      = newsys
    Type      = mamdani
    NumInputs = 0
    InLabels  =
    NumOutputs = 0
    OutLabels =
    NumRules  = 0
    AndMethod = min
    OrMethod  = max
    ImpMethod = min
    AggMethod = max
    DefuzzMethod = centroid
ans = [newsys]
```

(10) mf2mf 函数

功能：在隶属度函数之间进行参数变换。

格式：outParams=mf2mf(inParams, inType, outType)

说明：mf2mf 函数可根据隶属度函数的参数集，将一种隶属度函数变换成另一种，原则上，mf2mf 函数在新旧隶属度函数的对称点上匹配。这种变换偶尔也会导致信息的丢失，因此如果再将其变换回原来的隶属度函数类型时，则可能会与原隶属度函数不一致。

mf2mf 的输入变量如下。

inParams: 要变换的隶属度函数的参数。

inType: 代表要变换的隶属度函数类型的字符串。

outType: 代表要变换成新隶属度函数名的字符串。

【例 6-15】mf2mf 函数示例。

```
x=0:0.1:6;
mf1=[1 2 3];
mf2=mf2mf(mf1,'gbellmf','trimf');
plot(x,gbellmf(x,mf1),x,trimf(x,mf2));
```

运行程序，效果如图 6-17 所示。

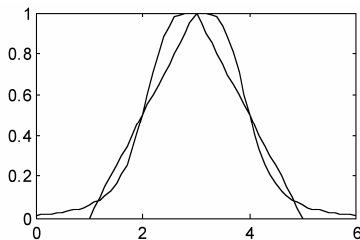


图 6-17 隶属度函数之间的变换

(11) readfis 函数

功能：从磁盘中装入 FIS。

格式：fismat=readfis('filename')

说明：从磁盘的 filename.fis 文件中读取模糊推理系统，并存于工作空间中。fismat= readfis (不带输入变量) 将打开读取文件的对话框，以便输入文件名及其路径。

【例 6-16】readfis 函数示例。

```
fismat=readfis('tipper');
getfis(fismat)
```

输出如下：

```
Name      = tipper
Type      = mamdani
NumInputs = 2
InLabels  =
    service
    food
NumOutputs = 1
OutLabels =
    tip
NumRules = 3
AndMethod = min
OrMethod  = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans = tipper
```

(12) plotfis 函数

功能：绘图表示 FIS。

格式：plotfis(fismat)

说明：plotfis 函数可绘制出 FIS 结构（由 fismat 指定）的框图。输入及其隶属度函数在左边，输出及其隶属度函数绘制在右边。

【例 6-17】plotfis 函数示例。

```
>> a=readfis('tipper')
plotfis(a)
```

运行程序得到 tipper 模糊推理系统，如图 6-18 所示。

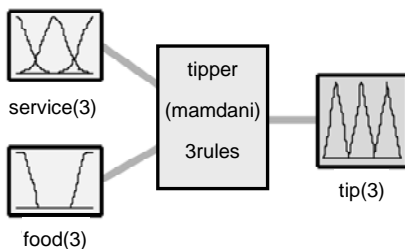


图 6-18 FIS 绘图表示

(13) plotmf函数

功能：绘制出给定变量的所有隶属度函数。

格式：plotmf(fismat,'varType',varIndex)

说明：plotmf 函数可绘制出 FIS 中给定变量的所有隶属度函数，其中 fismat 指定 FIS 结构，varType 指定变量类型（可取 input 或 output），varIndex 指定变量的序号。这一函数还可以与 MATLAB 的 subplot 配合使用。

【例 6-18】plotmf 函数示例。

```
a=readfis('tipper');
plotmf(a,'input',1)
```

执行程序得到隶属度函数曲线，如图 6-19 所示。

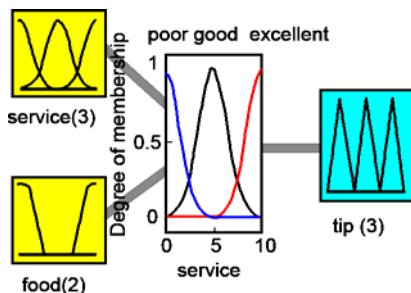


图 6-19 输入变量 1 的所有隶属度函数

(14) parsrule函数

功能：模糊规则解析。

格式：fis2=parsrule(fis,txtRuleList)

fis2=parsrule(fis,txtRuleList, ruleFormat)

fis2=parsrule(fis,txtRuleList,ruleFormat,lang)

说明：这一函数可对 MATLAB 工作空间中 FIS 变量定义的规则（由 txtRuleList 指定）进行解析，如果原来的 FIS 结构具有初始的规则，则它们将在新的结构中被取代。这里支持 3 种规则格式（由 ruleFormat 指定）：verbose（详细）、symbolic（符号）和 indexed（编号），其默认格式为 verbose。当使用语言变量 lang 时，规则按 verbosr 模式解析，并采用在 lang 中指定的关键字进行解析。语言只能取 English（英语）、Francais（法语）或 Deutsch（德语），在 English 中的关键字为：If、then、is、and、or 和 not。

【例 6-19】parsrule 函数示例。

```
a=readfis('tipper');
ruleT='If service is poor then tip is generous';
a1=parsrule(a,ruleT,'verbose');
showrule(a1)
ans =
1. If (service is poor) then (tip is generous) (1)
```

(15) rmmf函数

功能：从 FIS 中删除隶属度函数。

格式：fis=rmmf(fis,'varType',varIndex,'mf',mfIndex)

说明：其中 `fis` 指定 FIS 结构，`varIndex` 指定变量序号，`varType` 指定变量类型（可取 `input` 或 `output`），`mfIndex` 指定要删除的隶属度函数的序号，字符串 `mf` 表示要删除的是隶属度函数。

【例 6-20】`rmmf` 函数示例。

```
a=newfis('my_fun');
a=addvar(a,'input','temperature',[0 100]);
a=addvar(a,'input',1,'cold','trimf',[0 30 60]);
getfis(a,'input',1)
    Name =      temperature
    NumMFs =      0
    MFLabels =
    Range =      [0 100]
```

输出如下：

```
ans =
    Name: 'temperature'
    NumMFs: 0
    range: [0 100]
```

如果输入：

```
b=rmmf(a,'input',1,'mf',1);
getfis(b,'input',1)
```

得到输出如下：

```
Name =      temperature
    NumMFs =      0
    MFLabels =
    Range =      [0 100]
```

（16）`rmvar` 函数

功能：从 FIS 中删除变量。

格式：`[fis2, errorStr]=rmvar(fis, 'varType', varIndex)`

`fis2=rmvar(fis, 'varType', varIndex)`

说明：`fis2=rmvar(fis, 'varType', varIndex)` 可从 FIS 中删除变量，其中 `fis` 指定 FIS 结构，`varIndex` 指定要删除的变量序号，`varType` 指定变量类型（可取 `input` 或 `output`）。

`[fis2, errorStr]=rmvar(fis, 'varType', varIndex)` 可在 `errorStr` 中得到错误信息。

这一函数还可以自动调整规则列表，以便使它与当前的变量数一致。但必须在删除变量前，从 FIS 中删除要删除变量的所有规则，不能删除在规则列表中当前正在使用的模糊变量。

【例 6-21】`rmvar` 函数示例。

```
a=newfis('my_fun');
a=addvar(a,'input','temperature',[0 100]);
getfis(a)
```

输出如下：

```
Name      = my_fun
Type      = mamdani
```

```

NumInputs = 1
InLabels   =
    temperature
NumOutputs = 0
OutLabels  =
NumRules   = 0
AndMethod  = min
OrMethod   = max
ImpMethod  = min
AggMethod  = max
DefuzzMethod = centroid

```

如果输入:

```

b=rmvar(a,'input',1);
getfis(b)

```

得到输出如下:

```

Name      = my_fun
Type      = mamdani
NumInputs = 0
InLabels  =
NumOutputs = 0
OutLabels =
NumRules  = 0
AndMethod = min
OrMethod  = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans = my_fun

```

(17) showfis函数

功能: 显示带注释的 FIS。

格式: showfis(fismat)

说明: showfis(fismat)可显示出 FIS 结构 fismat, 从而更容易观察 FIS 结构各个域的重要性及其内容。

【例 6-22】 showfis 函数示例。

```

>> a=readfis('tipper');
showfis(a)

```

输出如下:

```

1. Name      tipper
2. Type      mamdani
3. Inputs/Outputs [2 1]
4. NumInputMFs [3 2]
5. NumOutputMFs 3
6. NumRules   3

```

7. AndMethod	min
8. OrMethod	max
9. ImpMethod	min
10. AggMethod	max
11. DefuzzMethod	centroid
12. InLabels	service
13.	food
14. OutLabels	tip
15. InRange	[0 10]
16.	[0 10]
17. OutRange	[0 30]
18. InMFLabels	poor
19.	good
20.	excellent
21.	rancid
22.	delicious
23. OutMFLabels	cheap
24.	average
25.	generous
26. InMFTypes	gaussmf
27.	gaussmf
28.	gaussmf
29.	trapmf
30.	trapmf
31. OutMFTypes	trimf
32.	trimf
33.	trimf
34. InMFParams	[1.5 0 0 0]
35.	[1.5 5 0 0]
36.	[1.5 10 0 0]
37.	[0 0 1 3]
38.	[7 9 10 10]
39. OutMFParams	[0 5 10 0]
40.	[10 15 20 0]
41.	[20 25 30 0]
42. Rule Antecedent	[1 1]
43.	[2 0]
44.	[3 2]
42. Rule Consequent	1
43.	2
44.	3
42. Rule Weight	1
43.	1
44.	1
42. Rule Connection	2
43.	1
44.	2

(18) setfis函数

功能：设置模糊系统的特性。

格式：a=setfis(a,'fisPropname','newfisProp')

a=setfis(a,'varType','varIndex','varPropname','newvarProp')

a=setfis(a,'varType','varIndex','mf','mfIndex','mfPropname','newmfProp')

说明：根据要设置的 FIS 特性不同，setfis 命令可有 3 个、5 个或 7 个输入变量，这些变量含义如下。

a：工作空间中的 FIS 变量名。

varType：变量类型的字符串（可取 input 或 output）。

varIndex：输入/输出变量序号。

mf：使用 7 个输入变量调用 setfis 时不可默认的变量，用于指示要设置隶属度函数的特性。

mfIndex：所选变量的隶属度函数序号。

fisPropname：表示 FIS 特性的字符串，这里可取 Name、Type、AndMethod、OrMethod、ImpMethod、AggMethod 和 DefuzzMethod。

newfisProp：描述 FIS 特性或方法的字符串。

varPropname：表示变量域名的字符串，这里可取 Name 或 Range。

newvarProp：当变量域名为 Name 时，这一部分为要设置的变量名的字符串；当变量域名为 Range 时，这一部分为该变量范围的阵列。

mfPropname：表示隶属度函数域名的字符串，可取 Name、Type 或 Params。

newmfProp：当变量域名为 Name 或 Type 时，这一部分为要设置的隶属度函数域名或类型；当变量域名为 Params 时，这一部分为参数范围的阵列。

【例 6-23】setfis 函数示例。

① 以 3 个自变量调用 setfis，方法如下。

```
a1=readfis('tipper');
a2=setfis(a,'name','singing');
getfis(a2,'name')
```

输出如下：

```
ans = singing
```

② 以 5 个自变量调用 setfis，则可以修改 2 个变量的特性，方法如下。

```
a3=setfis(a1,'input',1,'name','hello');
getfis(a3,'input',1,'name')
```

输出如下：

```
ans = hello
```

③ 以 7 个自变量调用 setfis，则可以修改几个隶属度函数的特性，方法如下。

```
a4=setfis(a1,'input',1,'mf',2,'name','match');
getfis(a4,'input',1,'mf',2,'name')
```

输出如下：

```
ans = match
```

(19) writefis函数

功能：将 FIS 结构保存到磁盘文件中。

格式：writefis(fismat)

writefis(fismat,'filename')

writefis(fismat,'filename','dialog')

说明：writefis 可将 MATLAB 工作空间中的 FIS 结构变量 fismat 保存到磁盘文件中。

writefis(fismat)可打开一个对话框，以输入变量的文件名及其路径。writefis(fismat,'filename')可直接指定文件名 filename.fis，这时不会出现对话框，文件保存在当前目录中。writefis (fismat,'filename','dialog')可打开对话框，并以 filename.fis 为默认文件名。

【例 6-24】writefis 函数示例。

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'myfile')
```

(20) showrule函数

功能：显示 FIS 规则。

格式：showrule(fis)

showrule(fis, indexList)

showrule(fis, indexList, format)

showrule(fis, indexList, format, Lang)

说明：showrule 可显示出 FIS 系统的规则，它可有 1~4 个输入变量，fis 为 FIS 的结构变量名，indexList 为要显示规则的序号向量，format 用于指定规则显示的格式，Lang 用于指定显示规则的语言，规则的显示可采用三种格式：verbose（详细）、symbolic（符号）和 indexed（编号）。

【例 6-25】showrule 函数示例。

```
>> a=readfis('tipper');
showrule(a,1)
ans =
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
>> showrule(a,2)
ans =
2. If (service is good) then (tip is average) (1)
>> showrule(a,[3 1],'symbolic')
ans =
3. (service==excellent) | (food==delicious) => (tip=generous) (1)
1. (service==poor) | (food==rancid) => (tip=cheap) (1)
>> showrule(a,[3 1],'indexed')
ans =
3 2, 3 (1) : 2
1 1, 1 (1) : 2
```

3. 先进技术

(1) anfis函数

功能: Sugeno 型 FIS 的训练程序。

格式: `[fismat, error, stepsize]=anfis(trnData)`

`[fismat1, error1, stepsize]=anfis(trnData, fismat)`

`[fismat1, error1, stepsize]=anfis(trnData, fismat, trnOpt, dispOpt)`

`[fismat1, error1, stepsize, fismat2, error2]=anfis(trnData, fismat, trnOpt, dispOpt, chkData, optMethod)`

说明: 这是 Sugeno 型模糊推理系统 (FIS) 的主要训练程序, `anfis` 利用一种混合学习算法来确定 Sugeno 型 FIS 的参数, 它结合最小二乘法与 BP 梯度下降法给出数据集训练, 从而确定 FIS 隶属度函数的参数。调用 `anfis` 时还可以使用一个可选参数检测模型的有效性, `anfis` 函数的变量如下。

trnData: 训练的数据集合名称, 这个矩阵的最后一列为输出数据, 其他列为输入数据。

fismat: FIS 名称, 用于为 `anfis` 提供训练隶属度函数的一组初值, 当这一选项为默认时, `anfis` 将使用 `genfis1` 产生 FIS, 这时 FIS 只含一个变量调用 `anfis`, 默认的 FIS 为具有两个高斯型的隶属度函数。如果 `fismat` 为单个数值 (或向量), 则将这个数值作为隶属度函数的数值 (或者将向量的值作为相应输入的隶属度函数的数值), 在这种情况下, 在开始训练之前, 将这两个变量传递给 `genfis1` 函数以产生有效 FIS 结构。

trnOpt: 训练的选项向量, 当向量选项为 NaN 时, 则取其默认值。各选项的默认值分别如下。

- `trnOpt(1)`: 训练批数 (默认值为 10)。
- `trnOpt(2)`: 训练误差目标 (默认值为 0)。
- `trnOpt(3)`: 初始步长 (默认值为 0.01)。
- `trnOpt(4)`: 步长减量比 (默认值为 0.9)。
- `trnOpt(5)`: 步长增量比 (默认值为 1.1)。

dispOpt: 显示选项向量, 用于指定训练中的显示信息, 默认值为 1, 表示显示相应的信息; 其值为 0, 表示不显示信息。当显示选项为 NaN 时, 表示取默认值。这些选项包括如下项。

- `dispOpt(1)`: ANFIS 信息, 如输入和输出隶属度函数的数量 (默认值为 1)。
- `dispOpt(2)`: 误差 (默认值为 1)。
- `dispOpt(3)`: 参数更新步长 (默认值为 1)。
- `dispOpt(4)`: 最终结果 (默认值为 1)。

chkData: 隶属度有效性检查的数据集名称, 这里的数据集是一个与训练数据集具有相同格式的矩阵。

optMethod: 隶属度函数训练中的可选最优化方法, 其中 1 表示混合方法, 0 表示 BP 方法。默认时为混合方法, 即最小二乘法与 BP 的组合。当该选项为非 0 值时就取其默认方法。

一旦达到设计的训练批数或者达到训练的误差目标, 则停止训练过程。训练结束后输出变量如下。

format1: 根据最小训练误差准则而得到的 FIS 结构。

error1 和 error2: 分别表示训练数据和检验数据的均方根误差。

format2: 根据最小检验误差准则而得到的 FIS 结构。

(2) genfis1 函数

功能: 从未加聚类的数据中产生 FIS 结构。

格式: fismat=genfis1(data)

fismat=genfis1(data,numMFs,inMFTType, outMFTType)

说明: genfis1 可为训练 ANFIS 产生 Sugeno 型 FIS 结构的初值 (隶属度函数参数的初值), genfis1(data, numMFs, inMFTType, outMFTType) 是利用在未加聚类的数据中进行栅格分区的方法, 可从训练数据集 data 中产生 FIS 结构, 其输入变量如下。

Data: 训练数据向量, 除了最后一列为输出数据外, 其余列均表示输入数据。

numMFs: 表示向量, 其每个元素表示输入的隶属度函数的数量。如果每个输入的隶属度函数数量相同, 则只需要输入标题值。

inMFTType: 表示字符串阵列, 其每一行指定一个输入的隶属度函数类型, 如果隶属度函数类型相同, 则 inMFTType 变成一维的字符串。

outMFTType: 指定输入隶属度函数类型的字符串。由于只能采用 Sugeno 型系统, 因此系统只能有一个输出, 其类型为 linear 或 constant。

与输出相对应的隶属度函数数目等于由 genfis1 产生的规则数。隶属度函数 numMFs 的默认值为 2, 输入/输出隶属度函数类型的默认值为 gbellmf。

(3) genfis2 函数

功能: 利用减法聚类从数据中产生 FIS 结构。

格式: fismat=genfis2(Xin, Xout, radii)

fismat=genfis2(Xin, Xout, radii, xBounds)

fismat=genfis2(Xin, Xout, radii, xBounds, options)

说明: 在给定输入和输出数据的情况下, genfis2 函数可利用模糊减法聚类产生 FIS 结构。当只有一个输出时, genfis2 通常在数据上实现减法聚类来产生训练 anfis 的初始 FIS, 它是通过提取一组规则, 对数据进行建模来完成的。规则提取方法先利用 subclust 函数确定规则数和隶属度函数, 然后利用线性最小二乘法估计每条规则的方程, 由此得到 FIS 结构, 其模糊规则覆盖了特征空间。

genfis2 函数的输入变量如下。

Xin: 表示一个矩阵, 其每一行为数据点的输入值。

Xout: 表示一个矩阵, 其每一行为数据点的输出值。

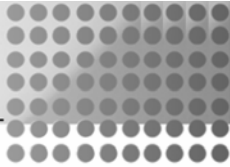
radii: 表示一个向量, 用于指定每个数据维上聚类中心的范围 (设数据在单位超立方体内), 例如, 数据维为三 (设 Xin 有两列, Xout 有一列), 则 radii=[0.5 0.4 0.3] 指定了第 1、2、3 个数据维 (即 Xin 的第一列、Xin 的第二列和 Xout 的列) 的波动范围分别为数据空间宽度的 0.5 倍、0.4 倍和 0.3 倍。如果 radii 为标量, 则所有数据维具有相同的倍数, 也就是说, 每个聚类中心都具有一个以给定值为半径的球形波动领域。

xBounds: 为 2×n 的可选矩阵, 用于指定将 Xin 和 Xout 中的数据映射到单位超立方体的方法。xBounds 的第一行和第二行分别包含维数缩放时的最小值和最大值。

options: 表示可选向量, 用于指定算法参数, 这些参数在 subclust 函数在线帮助中解释。当没有指定这个变量时, 采用默认值。

(4) fcm 函数

功能: 模糊 C 均值聚类。



格式: [center, U, obj_fcn]=fcm(data, cluster_n)

说明: 表示在给定的数据集合上应用模糊 C 均值聚类方法。

fcm 函数的输入变量如下。

data: 要聚类的数据集合, 每一行为一个取样数据点。

cluster_n: 聚类数 (大于 1)。

fcm 函数的输出变量如下。

center: 最终的聚类中心矩阵, 其每一行为聚类中心的坐标值。

U: 最终的模糊分区矩阵 (或者称为隶属度函数矩阵)。

obj_fcn: 在迭代过程中的目标函数值。

fcm(data,cluster_n,options)可利用控制聚类参数的附加参量 options, 以引入停止准则和设置迭代信息的显示。

- options(1): 分区矩阵 U 的指数 (默认值为 2.0)。
- options(2): 最大的迭代次数 (默认值为 100)。
- options(3): 最小的改善量 (默认值为 10^{-5})。
- options(4): 迭代过程显示信息 (默认值为 1)。

如果在 options 中出现 NaN 值, 则取其默认值。

如果达到最大的迭代次数或者两次相继迭代之间的目标函数改善量小于指定的最小改善量, 则聚类过程结束。

6.3 MATLAB模糊逻辑工具箱命令函数应用示例

【例 6-26】许多工业控制过程都可以等效成二阶环节。设计典型二阶环节的模糊控制器, 使系统输出尽快跟随系统的输入。

$$H(s) = \frac{20}{1.6s^2 + 4.4s + 1}$$

若设系统输入 $R=1.5$, 系统输出误差为 e , 误差导数为 \dot{e} (程序中记作 de), 则可根据系统输出的误差和误差导数设计出模糊控制器 (FC)。FC 的输入为 e 和 \dot{e} 的模糊量: NB (负大)、NS (负小)、ZR (零)、PS (正小) 和 PB (正大)。FC 的输出为控制 u 的模糊量: NB (负大)、NS (负小)、ZR (零)、PS (正小) 和 PB (正大)。FC 的模糊推理规则, 如表 6-1 所示。

表 6-1 FC 的模糊推理规则

<div>u \ e</div> <div>de</div>	NB	NS	ZR	PS	PB
NB	PB	PB	PB	PS	ZR
NS	PB	PB	PB	ZR	ZR
ZR	PS	PS	ZR	ZR	NS
PS	PS	ZR	ZR	NS	NS
PB	ZR	ZR	NS	NS	NB



本例 MATLAB 代码如下:

```
% 典型二阶系统的模糊控制
% 控制对象建模
num=20;
den=[1.6 4.4 1];
[a1 b c d]=tf2ss(num,den);
x=[0;0];
t=0.01;h=t;
n=250;
R=1.5*ones(1,n); % 参数输入
% 定义输入/输出变量与隶属度函数
a=newfis('Simple');
a=addvar(a,'input','e',[-6,6]);
a=addmf(a,'input',1,'NB','trapmf',[-6,-6,-5,-3]);
a=addmf(a,'input',1,'NS','trapmf',[-5,-3,-2,0]);
a=addmf(a,'input',1,'ZR','trimf',[-2,0,2]);
a=addmf(a,'input',1,'PS','trapmf',[0,2,3,5]);
a=addmf(a,'input',1,'PB','trapmf',[3,5,6,6]);
a=addvar(a,'input','de',[-6,6]);
a=addmf(a,'input',2,'NB','trapmf',[-6,-6,-5,-3]);
a=addmf(a,'input',2,'NS','trapmf',[-5,-3,-2,0]);
a=addmf(a,'input',2,'ZR','trimf',[-2,0,2]);
a=addmf(a,'input',2,'PS','trapmf',[0,2,3,5]);
a=addmf(a,'input',2,'PB','trapmf',[3,5,6,6]);
a=addvar(a,'output','u',[-3,3]);
a=addmf(a,'output',1,'NB','trapmf',[-3,-3,-2,-1]);
a=addmf(a,'output',1,'NS','trimf',[-2,-1,0]);
a=addmf(a,'output',1,'ZR','trimf',[-1,0,1]);
a=addmf(a,'output',1,'PS','trapmf',[0,1,2]);
a=addmf(a,'output',1,'PB','trapmf',[1,2,3,3]);
% 模糊规则矩阵
rr=[5 5 4 4 3;
    5 4 4 3 3;
    4 4 3 3 2;
    4 3 3 2 2;
    3 3 2 2 1];
rr=zeros(prod(size(rr)),3);
k=1;
for i=1:size(rr,1)
    for j=1:size(rr,2)
        r1(k,:)= [i,j,rr(i,j)];
        k=k+1;
    end
end
end
```

```

[r,s]=size(r1);
r2=ones(r,2);
rulelist=[r1,r2];
a=addrule(a,rulelist);
%模糊控制系统仿真
e=0;
de=0;
ke=30;
kd=20;
ku=1;
for k=1:n
    e1=ke*e;
    de1=kd*de;
    if e1>=6;
        e1=6;
    elseif e1<=-6
        e1=-6;
    end
    if de1>=6
        de1=6;
    elseif de1<=-6
        de1=-6;
    end
    %模糊推理, 计算被控对象输入
    ni=[e1 de1];
    u=ku*evalfis(ni,a);
    uu(1,k)=u;
    %控制作用于被控系统, 计算系统输出
    k0=a1*x+b*u;
    k1=a1*(x+h*k0/2)+b*u;
    k2=a1*(x+h*k1/2)+b*u;
    k3=a1*(x+h*k2/2)+b*u;
    x=x+(k0+2*k1+2*k2+k3)*h/6;
    y=c*x+d*u;
    yy(1,k)=y;
    %计算系统输出误差及误差变化率
    e1=e;
    e=y-R(1,k);
    de=(e-e1)/t;
end
%模糊控制输出曲线
kk=[1:n]*t;
figure(1);
plot(kk,r,'r',kk,yy,'g');
grid on

```

运行程序，得到系统阶跃响应曲线，如图 6-20 所示。

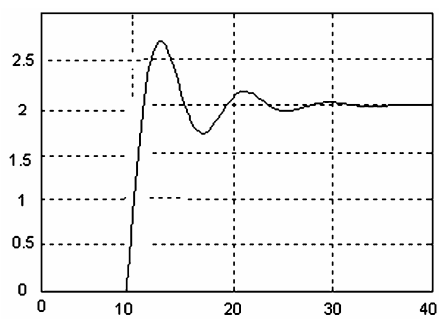


图 6-20 模糊控制系统输出

第7章 MATLAB在人工神经网络中的应用

7.1 人工神经网络介绍

人工神经网络是现代人工智能中一个重要的分支，它模拟人体神经元的工作原理，并通过数学模型的抽象简化发展而成。人工神经网络应用于现代工业控制各个领域，可以应用于系统参数辨识、模式识别、分类问题等。

经过几十年的发展，目前神经网络已经形成了数十种网络，包括多层感知器、Kohonen 自组织特性映射、Hopfield 网络、自适应共振理论、ART 网络、RBF 网络，以及近年来出现的小波神经网络和概率神经网络等。这些网络，由于结构不同，应用范围也不同。但它们都有如下的共同特点。

(1) 学习能力

学习能力是神经网络的重要表现，即通过训练可抽象出训练样本的主要特征，表现出强大的自适应能力。

(2) 分布式

神经网络中，信息分散分布在神经元的连接权上，只有将许多神经元的权值联合起来才能发挥作用，个别神经元受到损坏，并不会影响整体性能，这使得神经网络表现了强大的鲁棒性能。在输入信号受到一定扰动时，输出不会有较大的畸变。并且，这种信息分布的特性，使得经过训练后的神经网络有强大的联想能力。

(3) 并行性

各个神经元在处理信息时是各自独立的，它们分别接受输入，作用后产生输出，是一种并行的处理机制。

(4) 非线性

神经网络可以有效地实现输入空间到输出空间的非线性映射，对于大部分无模型的非线性系统，神经网络都能很好地进行模拟。因此，神经网络是对非线性系统进行研究的重要工具。

当然，神经网络并非完物，也存在一些问题，比如收敛速度慢、易陷入局部极小。这是因为在该算法中的网络的权值依赖误差函数的一阶导数来进行修正，即误差的负梯度方向，当求解空间存在多个局部极小的时候，一旦随机产生的初始网络权值设置不当就可能陷入局部极小，即使初始网络权值已经在全局最优解旁边，当解的周围平坦，导致梯度变化很小时，收敛速度会特别小。

下面分别对几种著名的神经网络进行介绍。

7.2 感知器

感知器 (Perceptron) 是由美国科学家 F.Rosenblatt 于 1957 年提出的，其目的是为了模拟人脑的感知和学习能力。感知器是最早提出的一种神经网络模型。它特别适合于简单的模式分类问题，如线性可分的形式。

7.2.1 感知器原理

感知器的神经元的激发函数是符号函数，最简单的感知器的输出为 0 或 1，即将输入向量分成了两个部分，用 0 和 1 来区分。感知器神经元模型如图 7-1 所示。

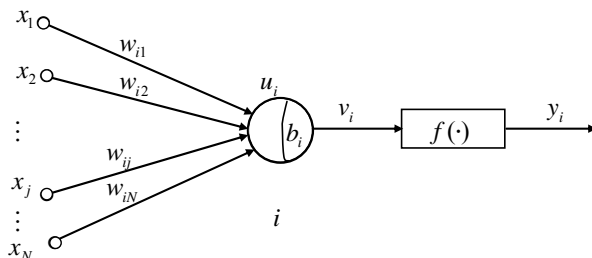


图 7-1 感知器神经元模型

其中， N 为输入向量的个数，图 7-1 中所示的感知器神经元模型中仅含单一的神经元，其神经元构成包括同输入向量的权重系数 w_{i1} 和阈值 b_i 。当感知器神经元模型中包含多个神经元时，只需要将多个神经元串联，并形成网络拓扑结构，同时对应于 v 个神经元输出，那么第 i 个神经元的输出为

$$v_i = \sum_{k=1}^N w_{ik} x_k + b_i$$

感知器的输出函数由符号函数阈值单元构成，使用 `hardlim` 函数实现，其功能与 `sign` 函数相同，当输入 $v_i \geq 0$ 时， $y_i = 1$ ，当 $v_i \leq 0$ 时， $y_i = 0$ ，那么第 i 个神经元经过符号阈值后的输出为

$$y_i = f(v_i) = \text{hardlim}(wx+b)$$

在 MATLAB 命令行窗口中输入：

```
>> x=-5:0.01:5;
>> y=hardlim(x);
>> plot(x,y)
>> axis([-5 5 -0.2 1.2]);
```

感知器阈值函数 `hardlim` 曲线如图 7-2 所示。

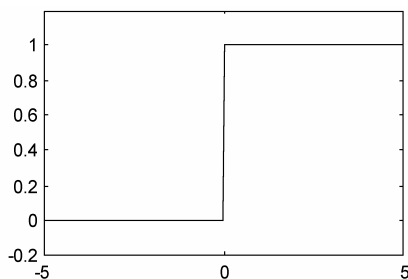
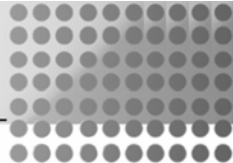


图 7-2 感知器阈值函数 `hardlim` 曲线



7.2.2 感知器相关函数

1. newp函数

功能：构建感知器模型函数。

格式：net=newp(PR,S,TF,LF)

说明：函数参数解析如下。

PR: $R \times 2$ 的输入向量最大值和最小值构成的矩阵。

S: 神经元的个数。

TF: 传输函数设置, 为 hardlim 函数或者 hardlims 函数, 默认为 hardlim 函数。

LF: 学习函数设置, 为 learnp 函数或者 learnpn 函数, 默认为 learnp 函数。

net: 生成的感知器网络。

【例 7-1】生成具有 1 个神经元的神经网络。

在命令窗口中输入：

```
>> %输入向量
P=[1.24 1.36 1.38 1.38 1.38 1.4 1.48 1.54 1.56 1.56 1.14 1.18 1.2 1.26 1.28 1.3;
    1.72 1.74 1.64 1.82 1.9 1.7 1.82 1.82 2.08 1.78 1.96 1.86 2.0 2.0 2.0 1.96];
%目标向量
T=[1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0];
net=newp([0 3;0 3],1);
```

查看工作窗口中的变量, 可以得到:

```
>> whos
  Name      Size      Bytes  Class
  P         2x16      256    double array
  T         1x16      128    double array
  net       1x1      18543   network object
Grand total is 679 elements using 18927 bytes
```

net 的变量属性是网络对象的结构体, 可以输入 net 后查看 net 结构体的相关信息。

```
>> inputweights=net.inputweights{1}
inputweights =
    delays: 0
    initFcn: 'initzero'      %网络权重初始化函数
    learn: 1
    learnFcn: 'learnp'      %默认的学习函数
    learnParam: []
    size: [1 2]             %神经网络的大小
    userdata: [1x1 struct]
    weightFcn: 'dotprod'     %权重函数、点积函数
```

通过以下命令查看神经网络各层权重以及神经元阈值情况。

```
>> net.IW{1}    %输入层神经元权重
ans =
    0    0
>> net.LW{1}    %隐层神经元权重
ans =
    []
>> net.b{1}     %神经元阈值
ans =
    0
```



2. adapt函数

功能：对神经网络进行自适应训练。

格式：[net,Y, E, Pf, Af, tr]=adapt(net, P, T, Pi, Ai)

说明：对输入/输出参数解析如下。

P：训练样本数据，对于 n 组训练样本， R 个输入端，则 P 的数据格式为 $R \times n$ 的数组。

T：目标样本数据，相对应于 P 数据，则为 $1 \times n$ 的输出向量。

Pi：初始化输入层延迟条件。

Ai：初始化层延迟条件。

Y：神经网络的输出数据，数据大小格式同输入的目标样本数据为 $1 \times n$ 的向量。

E：神经网络的输出误差数据，大小为 $1 \times n$ 的向量，使用 sse 函数或者 mse 函数计算网络误差。

Pf：训练后网络输出层延迟条件。

Af：训练后网络层延迟条件。

tr：网络训练过程数据记录，包括 epochs 和 pref 数据。

对例 7-1 构建的感知器网络进行自适应训练，得到以下结果：

```
>> [net,Y,E,Pf,Af,tr]=adapt(net,P,T);
Y =
    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
E =
    0    0    0    0    0    0    0    0    0    0   -1   -1   -1   -1   -1    1
tr =
    timesteps: 1
           perf: 0.3750
>> mse(E) %均方差结果显示
ans =
    0.4000
>> sse(E) %误差平方和结果显示
ans =
    6
```

3. sim函数

功能：使用 sim 函数进行神经网络仿真。

格式：[Y, Pf, Af, E, perf]=sim(net, P, Pi, Ai, T)

[Y, Pf, Af, E, perf]=sim(net, {Q TS}, Pi, Ai, T)

[Y, Pf, Af, E, perf]=sim(net, Q, Pi, Ai, T)

说明：输入/输出参数的解析可以参考 adapt 函数中的解析。使用 sim 函数同样可以实现感知器网络的训练。

利用上文中建立的感知器神经元模型，训练新的样本数据：

```
>> p=[1.24 1.28 1.4;1.8 1.84 2.04];
>> a=sim(net,p)
a =
    1    1    1
```

4. train函数

功能：对网络样本数据进行训练。

格式：[net, tr, Y, E, Pf, Af]=train(net, P,T, Pi, Ai, VV, TV)

说明: train 函数中输入参数和输出参数同 adapt 函数、sim 函数基本相同,其含义参看 adapt 函数中的参数说明。两个不同的输入参数分别是 VV 和 TV,前者表示给定向量结构体,后者表示测试向量结构体。通过设置 net.trainParam 的参数可以设置最大的迭代次数、感知器神经元模型训练目标、训练过程显示的训练代数以及训练时间设置,用户可以设置这些训练参数,改变神经网络模型 net 的相关属性。

```
>> net.trainParam
ans =
           show: 25      %训练过程显示训练次数
    showWindow: 1
    showCommandLine: 0
           epochs: 100    %最大迭代次数
            goal: 0       %网络训练目标
            time: Inf     %训练时间设置
```

【例 7-2】使用 train 函数训练感知器网络。

(1) 首先,输入训练样本和训练目标样本,创建单神经元感知器网络,设置训练函数 train 的参数,并进行训练,单神经元感知器训练过程如图 7-3 所示。

```
%输入训练样本数据
P=[1.24 1.36 1.38 1.38 1.38 1.4 1.48 1.54 1.56 1.56 1.14 1.18 1.2 1.26 1.28 1.3;
    1.72 1.74 1.64 1.82 1.9 1.7 1.82 1.82 2.08 1.78 1.96 1.86 2.0 2.0 2.0 1.96];
%训练样本目标数据
T=[1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0];
%创建单神经元感知器网络
net=newp([0 3;0 3],1);
%训练函数 train 的参数设置
net.trainParam.epochs=499;
net.trainParam.goal=0;
net.trainParam.show=49;
%使用 train 函数训练单神经元感知器网络
[net,tr,Y,E,Pf,Af]=train(net,P,T);
```

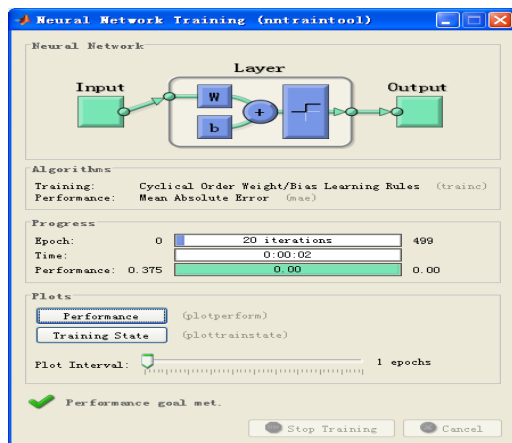


图 7-3 单神经元感知器训练过程

(2) 利用训练好的单神经元感知器模型仿真检验样本数据,结果如图 7-4 所示。

```
%使用测试样本数据并绘制分类图
figure
p=[1.24 1.28 1.4;1.8 1.84 2.04];
a=sim(net,p);
```

```

plotpv(p,a);
point=findobj(gca,'type','line');
set(point,'color','red');
hold on;
plotpv(P,T);
plotpc(net.IW{1},net.b{1});

```

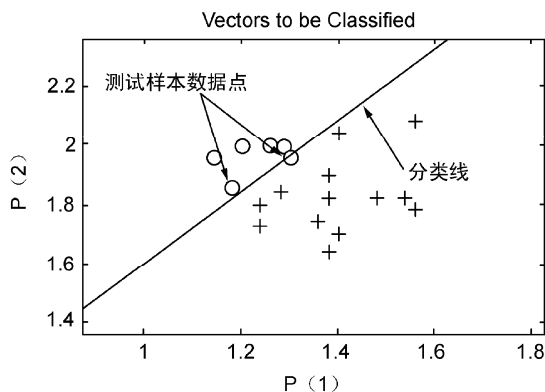


图 7-4 测试样本训练结果

7.2.3 感知器的MATLAB实现

【例 7-3】利用感知器实现三输入的与门功能。

利用感知器可以训练一个三输入的与门功能，其真值表如表 7-1 所示。在这里使用两种方法训练神经网络：一种方法是采用全样本数据训练，而检验样本为输入样本增加 0.2 的误差；另一种方法是采用部分样本数据训练，剩余真值表数据作为检验样本数据。

表 7-1 三输入与门真值表

样本数目	A	B	C	Y
1	0	1	0	0
2	0	1	1	0
3	0	0	1	0
4	1	0	0	0
5	1	0	0	0
6	1	0	1	0
7	0	1	0	0
8	0	1	1	0
9	1	1	1	1

训练方法一：采用全样本数据，检验样本增加 0.2。

```

>> %输入样本数据
P=[0 0 1 1 1 0 0 1;
    1 1 0 0 0 0 1 1 1;
    0 1 1 0 0 1 0 1 1];
%输入训练目标样本数据
T=[0 0 0 0 0 0 0 0 1];
%构建单神经元感知器网络

```

```

net=newp([ repmat([-1 2],3,1)],1);
%训练单神经元感知器网络
E=1;
while(sse(E))
    [net,Y,E,Pf,Af,tr]=adapt(net,P,T);
end
%训练样本数据增加 0.2 的扰动
p=P+0.2;
a=sim(net,p)
输出结果为:
a =
    0    0    0    0    0    0    0    0    1

```

可以看出仿真结果完全正确，训练好的单神经元感知器网络对输入样本数据具有一定的抗扰动的能力，如果扰动过大，那么感知器网络将带来较大的误差。如扰动增加到 0.3 时：

```

>> p=P+0.3;
a=sim(net,p)
a =
    0    0    0    1    1    1    0    0    1

```

从结果中可以看出，此时单神经元感知器网络训练误差比较大，有三组样本数据输出错误。所以，从该示例可以发现，神经网络具有一定的抗干扰能力，但扰动过大时，神经网络适应能力将下降。

训练方法二：采用部分样本数据作为训练样本。

```

>> %输入样本数据
P=[0 1 1 1 0 0;
    0 0 0 0 1 1;
    1 0 0 1 0 1];
%输入训练目标样本数据
T=[0 0 0 0 0 1];
%构建单神经元感知器网络
net=newp([ repmat([-1 2],3,1)],1);
%训练单神经元感知器网络
E=1;
while(sse(E))
    [net,Y,E,Pf,Af,tr]=adapt(net,P,T);
end
%检验样本数据
p=[0 1;0 1;1 0];
a=sim(net,p)

```

训练后输出结果为：

```

a =
    0    0

```

从结果中可以看到，训练后的单神经元感知器对检验数据输出错误。从这个示例来看，当利用单神经元感知器网络实现逻辑门功能时，由于训练样本的数据空间不是特别大，所以应该采用全训练样本集，以确保感知器网络的正确性。

7.3 线性神经网络

线性神经网络同感知器相似，是最简单的一种神经元网络，同感知器不同的是，线性神经

网络输出的激发函数为线性函数 `purelin`，而感知器模型的激发函数为符号函数 `hardlim`，因此感知器模型中只可能取 0 或 1，而线性神经网络输出的数值可以是任意数值，这一点也决定了线性神经网络同感知器应用范围的不同。

7.3.1 线性神经网络原理

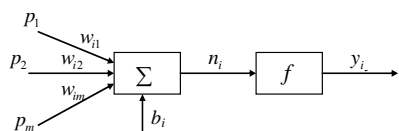


图 7-5 单个线性神经元模型的结构

线性神经网络由多个线性神经元模型构成，单个线性神经元模型的结构如图 7-5 所示。线性网络的激发函数为线性 `purelin` 函数，其输出可以为任意数值。

其中， m 为输入向量的个数，单个线性神经元同样由输入向量的权重系数 w_{i1} 和阈值 b_i 组成。当感知器神经元模型中包含多个神经元时，只需要将多个线性神经元串联，并形成网络拓扑结构，同时对应于 n 个神经元输出，那么第 i 个神经元的输出为

$$n_i = \sum_{k=1}^m w_{ik} x_k + b_i$$

感知器的输出函数由线性传递函数使用 `purelin` 函数实现。第 i 个神经元经过线性传递函数后的输出为

$$y_i = f(n_i) = \text{purelin}(wx+b)$$

在 MATLAB 命令行窗口中输入：

```
>> x=-5:0.01:5;
>> y=purelin(x);
>> plot(x,y);
>> title('purelin 函数曲线');
```

运行程序，效果如图 7-6 所示。

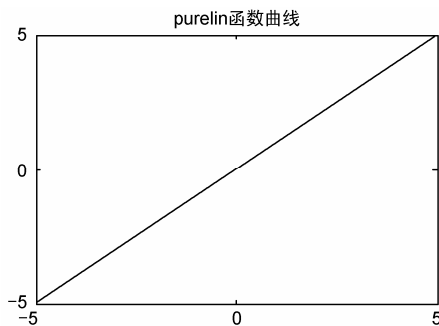


图 7-6 线性神经元 `purelin` 函数曲线

7.3.2 线性神经网络相关函数

1. `newlin` 函数

功能：构建线性神经网络函数。

格式：`net=newlin(PR,S, ID, LR)`

说明：其中 `PR` 为 $R \times 2$ 的输入向量最大值和最小值构成的矩阵，`S` 为输出向量的个数，`ID`

为输入延迟向量, LR 为学习速率, 学习速率可以使用 `maxlinlr` 函数进行计算, 通常学习速率越大, 网络训练时间越短, 但同时也导致学习过程不稳定。如果 `P` 为训练样本数据, 那么 `maxlinlr(P)` 返回一个不带阈值的线性所需要的最大学习率, 而 `maxlinlr(P, 'bias')` 返回一个带阈值的线性层所需要的最大学习率。

【例 7-4】使用 `newlin` 函数构建线性神经网络逼近函数。

$$y = \begin{cases} \cos(2\pi t), & 0 \leq t \leq 2 \\ \cos(4\pi t), & 2 < t \leq 4 \\ \cos(6\pi t), & 4 < t \leq 6 \end{cases}$$

(1) 首先, 产生输入训练样本和训练目标样本。

```
t1=0:0.01:2;
t2=2.01:0.01:4;
t3=4.01:0.01:6;
t=[t1 t2 t3];
T=[cos(t1*2*pi) cos(t2*4*pi) cos(t3*6*pi)];
T=con2seq(T); %将向量转化为序列 cell 向量
P=T; %输出向量等于给定输出向量
```

(2) 使用 `newlin` 函数构建线性神经网络, 并使用 `adapt` 函数训练网络。

```
lr=0.1;
delays=[1 2 3 4 5 6];
net=newlin(minmax(cat(2,P{:})),1,delays,lr);
[net,A,E]=adapt(net,P,T);
```

(3) 绘制网络输出信号和给定信号以及误差信号曲线。

```
plot(t,cat(2,P{:}),t,cat(2,A{:}),'r-+');
legend('给定输入信号','网络输出信号');
figure
plot(t,cat(2,E{:}));
title('误差曲线');
```

结果如图 7-7 及图 7-8 所示。

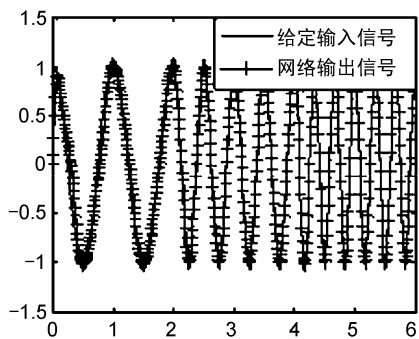


图 7-7 newlin 函数输入/输出信号曲线

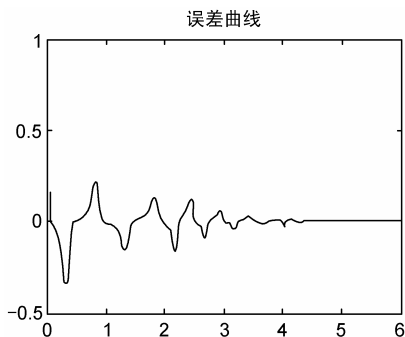


图 7-8 newlin 函数误差曲线

2. newlind函数

功能: 设计一个线性层。

格式: `net=newlind(P, T, Pi)`

说明：其中 P、T 分别是训练样本的输入矩阵和目标输出向量，Pi 是初始输入延时 cell 向量。
使用例 7-4 所示的示例演示 newlind 函数的使用。

(1) 首先，产生输入训练样本和训练目标样本。

```
t1=0:0.01:2;
t2=2.01:0.01:4;
t3=4.01:0.01:6;
t=[t1 t2 t3];
T=[cos(t1*2*pi) cos(t2*4*pi) cos(t3*6*pi)];
Q=length(T);
P=zeros(6,Q);
P(1,2:Q)=T(1,1:(Q-1));
P(2,3:Q)=T(1,1:(Q-2));
P(3,4:Q)=T(1,1:(Q-3));
P(4,5:Q)=T(1,1:(Q-4));
P(5,6:Q)=T(1,1:(Q-5));
P(6,7:Q)=T(1,1:(Q-6));
```

(2) 使用 newlind 函数设计线性层。

```
net=newlind(P,T);
a=sim(net,P);
plot(t,T,t,A,'k-+');
legend('给定输入信号','网络输出信号');
figure
plot(t,A-T);
title('误差曲线');
```

线性层网络函数逼近效果如图 7-9 及图 7-10 所示。

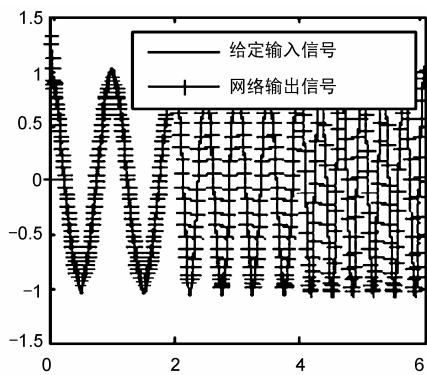


图 7-9 newlind 函数输入/输出信号曲线

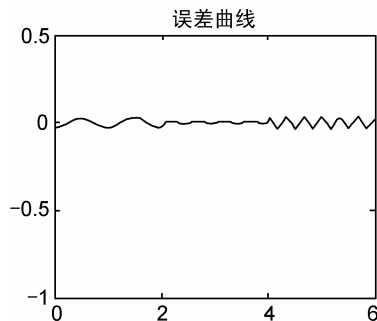


图 7-10 newlind 函数误差曲线

7.3.3 线性神经网络的MATLAB实现

线性神经网络的输出函数为线性传递函数 purelin，其输出可以为任意数值，因此线性神经网络应用范围非常广泛，可以应用于非线性系统拟合、函数逼近、系统辨识等方面。

【例 7-5】使用线性神经网络逼近非线性函数 $f(x) = 2x^6 + 3x^5 - 3x^3 + x^2 + 1$ ，绘制非线性函数并产生网络训练数据点。

```

x=-2:0.01:1;
y=2*x.^6+3*x.^5-3*x.^3+x.^2+1;
P=x(1:15:end);
T=y(1:15:end);
plot(x,y,P,T,'mo');
legend('逼近曲线','网络训练点');

```

其中， x 、 y 数据作为线性神经网络的训练数据点集，运行程序，效果如图 7-11 所示。
使用 `newlind` 函数训练线性神经网络：

```

Q=length(y);
r=zeros(6,Q);
r(1,2:Q)=y(1,1:(Q-1));
r(2,3:Q)=y(1,1:(Q-2));
r(3,4:Q)=y(1,1:(Q-3));
r(4,5:Q)=y(1,1:(Q-4));
r(5,6:Q)=y(1,1:(Q-5));
r(6,7:Q)=y(1,1:(Q-6));
net=newlind(r,y);
a=sim(net,r);
plot(x,y,x,a,'k-o');
legend('给定输入信号','网络输出信号');

```

运行程序，效果如图 7-12 所示。

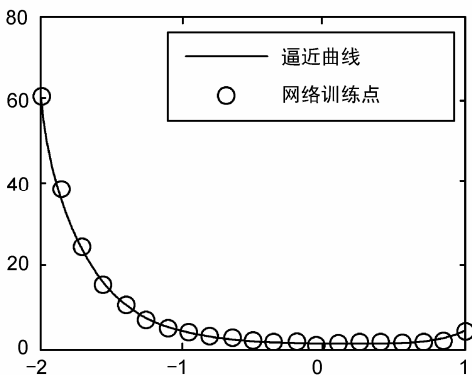


图 7-11 非线性逼近函数和网络训练点

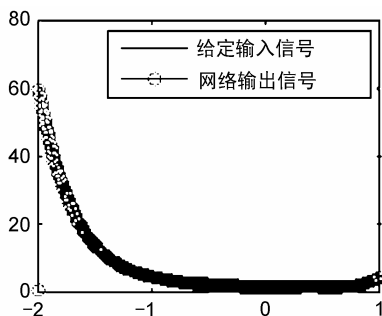


图 7-12 线性神经网络非线性曲线逼近结果

7.4 BP网络

在人工神经网络的实际应用中，使用最广泛的是采用反向传播算法的 BP 神经网络。BP 神经网络包含输入层、输出层和多个隐藏层，因此其结构复杂，适应性强，可以应用于各种函数逼近、模式识别、分类问题及数据压缩等。

7.4.1 BP网络原理

在如图 7-13 所示的通用神经元模型中， R 为输入向量的个数，通用神经元同样由输入向量的权重系数 w_{ji} 和阈值 b_j 组成。

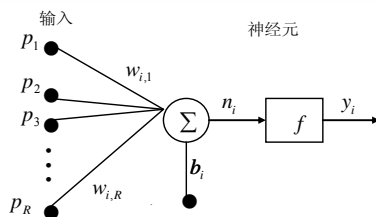


图 7-13 通用神经元模型

BP 神经网络中包含多个通用神经元模型，只需要将多个通用神经元串联，同时对应 n 个神经元输出，那么第 i 个神经元的输出为

$$n_i = \sum_{k=1}^R w_{ik} x_k + b_i$$

BP 网络的输出函数为任意函数，包括 `logsig` 函数、`tansig` 函数和 `purelin` 函数。第 i 个神经元经过任意传递函数后的输出为

$$n_i = \sum_{k=1}^R w_{ik} x_k + b_i$$

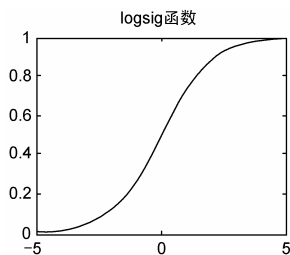
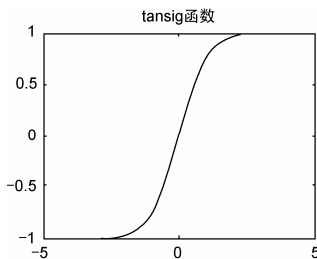
BP 网络的输出函数为任意函数，包括 `logsig` 函数、`tansig` 函数和 `purelin` 函数。第 i 个神经元经过任意传递函数后的输出为

$$y_i = f(n_i) = \text{logsig}(n_i) | \text{tansig}(n_i) | \text{purelin}(n_i)$$

在 MATLAB 命令行窗口中输入以下程序段：

```
>> x=-5:0.01:5;
>> y=logsig(x);
>> plot(x,y);
>> title('logsig 函数');
>> figure;
>> plot(x,y,'r');
>> title('tansig 函数');
```

运行程序后，得到普通神经元 `logsig` 函数曲线（见图 7-14）和 `tansig` 函数曲线（见图 7-15）。

图 7-14 `logsig` 函数曲线图 7-15 `tansig` 函数曲线

7.4.2 BP网络相关函数

1. `newff`函数

功能：创建一个反射传播算法的 BP 网络。

格式：`net=newff(PR,[S1 S2 ... SN], {TF1 TF2 ... TFN}, BTF, BLF, PF)`

说明：在输入参数中，PR 为 R 维的输入元素的 $R \times 2$ 最大值和最小值矩阵；SN 为第 N 层网络神经元的个数，共有 N 层；TFN 为第 N 层网络的转移函数，默认为 `tansig` 函数；BTF 为神经网络的训练函数，默认为 `trainlm` 函数；BLF 为神经网络权值/偏差的学习函数；PF 为性能评价函数，默认为 `mse` 函数。

下面利用蠼螋分类问题的数据，建立 BP 神经网络，使用不同的训练函数进行网络训练，如下所示。

```
%输入样本
P=[1.24 1.36 1.38 1.38 1.38 1.4 1.48 1.54 1.56 1.14 1.18 1.2 1.26 1.28 1.3;
    1.72 1.74 1.64 1.82 1.9 1.7 1.82 1.82 2.08 1.78 1.96 1.86 2.0 2.0 1.96];
%目标样本
T=[1 1 1 1 1 1 1 1 1 0 0 0 0 0 0];
%检验样本
p=[1.24 1.28 1.4;1.8 1.84 2.04];
%创建一个两层的 BP 网络，转换函数分别为 logsig 函数和 purelin 函数
net=newff(minmax(P),[5 1],{'logsig','purelin'});
a=sim(net,P)
```

没有经过训练的神经网络输出情况为：

```
a =
Columns 1 through 14
    0.7791    0.6919    0.2880    0.7572    0.4973    0.4501    0.3927    0.1612
Columns 9 through 15
    0.2634    0.7212    0.3096    0.6881    0.3677    0.3750    0.4116
```

可以发现，此时网络的输出与目标样本的误差非常大。

2. BP网络训练函数

(1) `traingd` 函数

`traingd` 函数使用梯度下降算法训练神经网络，通过 `net.trainParam` 可以查看 `traingd` 函数网络训练的相关参数。

```
>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingd');
>> net.trainParam
ans =

        show: 25
    showWindow: 1
showCommandLine: 0
        epochs: 1000      %最大的迭代次数
         time: Inf
         goal: 0          %训练目标
    max_fail: 6
         lr: 0.0100      %学习速率
    min_grad: 1.0000e-010
```

使用 `traingd` 函数进行蠼螋分类，在 MATLAB 命令窗口中输入：

```
net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingd');
net.trainParam.show=49;
```

```

net.trainParam.lr=0.25;
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)

```

traingd 函数训练过程如图 7-16 所示，测试样本检验结果如下：

```

a =
    0.6501    0.5219    0.5370

```

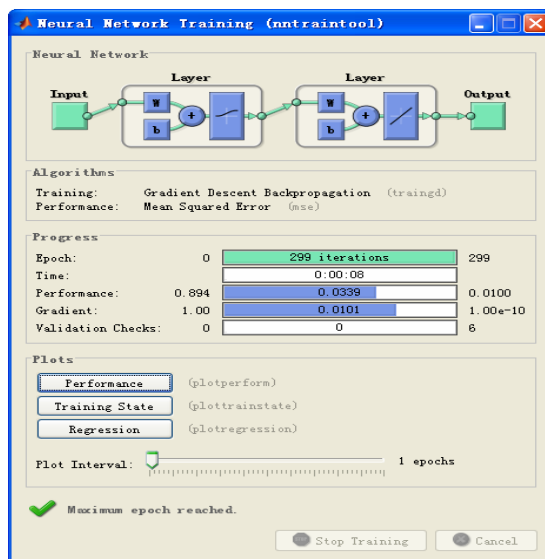


图 7-16 traingd 函数训练过程

(2) traingdm 函数

同 traingd 函数相似，traingdm 采用动量梯度下降法训练 BP 网络，通过设置 net.trainParam.mc 来设置动量因子的大小。

```

>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingdm');
>> net.trainParam
ans =

    show: 25
 showWindow: 1
showCommandLine: 0
    epochs: 1000
    time: Inf
    goal: 0
 max_fail: 6
    lr: 0.0100
    mc: 0.9000
 min_grad: 1.0000e-010

```

在 MATLAB 的命令行窗口中输入:

```
net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingdm');
net.trainParam.show=49;
net.trainParam.lr=0.1;
net.trainParam.mc=0.9;
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)
```

如图 7-17 所示为 traingdm 函数训练过程。训练后的网络对测试样本的输出为:

a =

0.7175 0.7159 0.4130

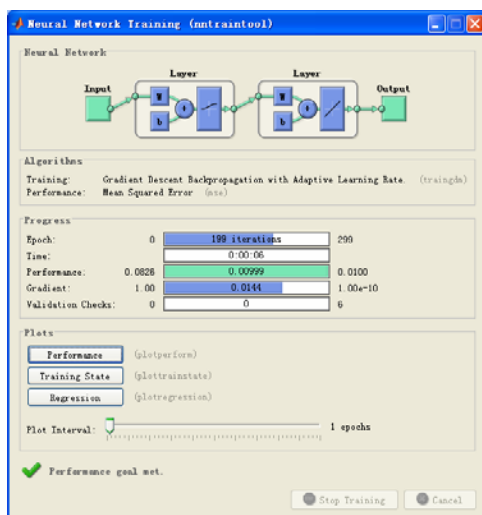


图 7-17 traingdm 函数训练过程

(3) traingda函数

当 BP 网络使用 traingd 函数和 traingdm 函数训练时, 学习速率在训练过程中保持恒定不变, 那么训练结果对学习速率的灵敏度大, 不同的学习速率对网络的训练结果影响大。如果学习速率过大, 那么网络将变得不稳定; 如果学习速率过小, 那么网络收敛速度慢, 训练时间大大加长。

因此, 对于给定训练样本和目标样本, 必须首先确定最优的学习速率。而 traingda 函数为变学习速率的网络训练算法, 可以有效地克服学习速率难以确定的缺点, 在训练过程中自适应改变学习速率的大小。

```
>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingda');
net.trainParam
ans =

    show: 25
showWindow: 1
```

```

showCommandLine: 0
    epochs: 1000
    time: Inf
    goal: 0
    max_fail: 6
    lr: 0.0100      %学习速率基值
    lr_inc: 1.0500  %学习速率增加率为 1.05
    lr_dec: 0.7000  %学习速率减小率为 0.7
    max_perf_inc: 1.0400
    min_grad: 1.0000e-010

```

在 MATLAB 命令行窗口中输入以下程序段：

```

net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingda');
net.trainParam.show=49;
net.trainParam.lr=0.1;
net.trainParam.lr_inc=1.05;
net.trainParam.lr_dec=0.85;
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)

```

其训练过程图不再给出，训练后的网络对测试样本的输出为：

```

a =
    0.6666    0.6822    0.2083

```

(4) traingdx 函数

结合了自适应改变学习速率和动量法，traingdx 函数和 traingda 函数完全相同，只是增加了一个动量因子参数 mc。

```

>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingdx');
net.trainParam
ans =

    show: 25
  showWindow: 1
showCommandLine: 0
    epochs: 1000
    time: Inf
    goal: 0
    max_fail: 6
    lr: 0.0100
    lr_inc: 1.0500
    lr_dec: 0.7000
    max_perf_inc: 1.0400
    mc: 0.9000
    min_grad: 1.0000e-010

```

在 MATLAB 命令行窗口中输入:

```
net=newff(minmax(P),[5 1],{'logsig','purelin'},'traingdx');
net.trainParam.show=49;
net.trainParam.lr=0.1;
net.trainParam.lr_inc=1.05;
net.trainParam.lr_dec=0.85;
net.trainParam.mc=0.9;
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)
```

训练后的网络对测试样本的输出为:

```
a =
    0.4818    0.5194    0.8511
```

(5) trainrp 函数

前面介绍的所有训练函数都采用梯度的变化量作为权重和阈值的变化依据,但是当函数在极值旁边非常平坦时,梯度的变化量非常小,那么权重和阈值的变化将非常小,过小的梯度变化一方面导致网络训练收敛速度非常慢;另一方面使网络无法达到训练性能要求。

trainrp 函数可以克服这方面的缺点,trainrp 函数使用误差梯度的方向来确定权重和阈值的变化,而误差梯度的大小对权重阈值的变化没有任何作用,权重阈值的变化量由另外两个参数 delt_inc 和 delt_dec 来确定。

trainrp 函数的训练参数设置为:

```
>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'trainrp');
net.trainParam
ans =

    show: 25
  showWindow: 1
showCommandLine: 0
    epochs: 1000
      time: Inf
      goal: 0
   max_fail: 6
  min_grad: 1.0000e-010
   delt_inc: 1.2000      %delta 值的增加率
   delt_dec: 0.5000      %delta 值的减小率
    delta0: 0.0700      %delta 初始值
   deltamax: 50          %最大的 delta 值
```

在 MATLAB 命令行窗口中输入:

```
net=newff(minmax(P),[5 1],{'logsig','purelin'},'trainrp');
net.trainParam.show=49;
```

```

net.trainParam.epochs=299;
net.trainParam.goal=0.01;
net.trainParam.delt_inc=1.5;
net.trainParam.delt_dec=0.8;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)

```

训练后的网络对测试样本的输出为:

```

a =
    0.2655    0.3715    0.7320

```

(6) traincgf函数

前面介绍的所有 BP 网络训练函数均采用负梯度方向,即在误差下降方向进行搜索,这样的搜索策略可以使目标函数即网络训练误差下降速率最快,但是却不能保证网络最快收敛,因此提出了共轭梯度算法,搜索方向为共轭梯度方向。

共轭梯度算法训练函数包括 `traincgf` 函数、`traincgp` 函数、`traincgb` 函数和 `trainscg` 函数。接下来,将介绍共轭梯度算法函数。

```

>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'traincgf');
net.trainParam
ans =
    show: 25
  showWindow: 1
showCommandLine: 0
    epochs: 1000
      time: Inf
      goal: 0
    max_fail: 6
    min_grad: 1.0000e-010
searchFcn: 'srchcha'
    scale_tol: 20
      alpha: 1.0000e-003
      beta: 0.1000
      delta: 0.0100
      gama: 0.1000
    low_lim: 0.1000
    up_lim: 0.5000
    maxstep: 100
    minstep: 1.0000e-006
      bmax: 26

```

在 MATLAB 命令行窗口中输入:

```

net=newff(minmax(P),[5 1],{'logsig','purelin'},'traincgf');
net.trainParam.show=49;

```

```
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
net.trainParam.searchFcn='srchbre';
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)
```

训练后的网络对测试样本的输出为:

```
a =
    0.7265    0.7366    0.0624
```

(7) traincgp函数

与 traincgf 函数相同, traincgp 函数采用共轭梯度算法进行 BP 网络训练, traincgp 函数的训练参数同 traincgf 完全相同, 具体参数设置见 traincgf 函数相关介绍。

在 MATLAB 命令窗口中输入:

```
net=newff(minmax(P),[5 1],{'logsig','purelin'},'traincgp');
net.trainParam.show=49;
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
net.trainParam.searchFcn='srchbre';
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)
```

训练后的网络对测试样本的输出为:

```
a =
    0.4882    0.5409    0.3872
```

读者可以参照 traincgf 和 traincgp 函数, 使用共轭梯度算法训练函数 traincgb 和 trainscg 进行 BP 网络训练, 体会函数间的差异。

(8) trainbfg函数

前面分别介绍了反向传播算法训练函数、线性搜索算法函数、共轭梯度算法训练函数。MATLAB 神经网络工具箱中还提供了拟牛顿训练函数 trainbfg 函数和 trainoss 函数。在这里, 主要介绍 trainbfg 训练函数, 读者自己可以仿照示例使用 trainoss 函数训练网络。trainbfg 函数训练参数通过以下指令获取:

```
>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'trainbfg');
net.trainParam
```

拟牛顿训练函数的相关参数同共轭梯度算法训练函数相似, 在此不再列出。在 MATLAB 命令行窗口中输入:

```
>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'trainbfg');
net.trainParam.show=49;
```



```

net.trainParam.epochs=299;
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)

```

训练后的网络对测试样本的输出为:

```

a =
    0.2770    0.2748    1.2451

```

(9) trainlm 函数

trainlm 函数同拟牛顿法相似, 使用一阶 Jacobian 矩阵近似计算二阶 Hessian 矩阵, 避免了网络训练误差的二阶导数计算。通过以下命令获取 trainlm 函数的参数设置:

```

>> net=newff(minmax(P),[5 1],{'logsig','purelin'},'trainlm');
net.trainParam
ans =
           show: 25
      showWindow: 1
 showCommandLine: 0
           epochs: 1000
              time: Inf
              goal: 0
          max_fail: 6
        mem_reduc: 1
        min_grad: 1.0000e-010    %网络训练停止的最小梯度
              mu: 1.0000e-003    %系数 u 的基值
          mu_dec: 0.1000    %系数 u 的减小因子
          mu_inc: 10    %系数 u 的增加因子
          mu_max: 1.0000e+010    %网络训练停止的最大 u 值

```

在 MATLAB 命令行窗口中输入:

```

net=newff(minmax(P),[5 1],{'logsig','purelin'},'trainlm');
net.trainParam.show=49;
net.trainParam.epochs=299;
net.trainParam.goal=0.01;
net.trainParam.mu_dec=0.1;
net.trainParam.mu_inc=7;
[net,tr]=train(net,P,T);
%回代检验
A=sim(net,P);
%测试样本检验
a=sim(net,p)

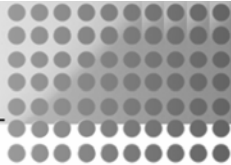
```

训练后的网络对测试样本的输出为:

```

a =
    0.8210    0.7992    0.2224

```



7.4.3 BP网络的MATLAB实现

在前面关于 BP 神经网络的函数逼近、分类问题中的应用都介绍了相关例子，BP 神经网络还应用于噪声去除问题。在 MATLAB 神经网络工具箱中，提供了 26 个大写字母的数据矩阵，利用 BP 神经网络，可以进行字符识别处理。

输入训练样本数据和测试样本：

```
%训练样本数据点
[AR,TS]=prprob;
A=size(AR,1);
B=size(AR,2);
C2=size(TS,1);
%测试样本数据点
CM=AR(:,13)
noisyCharM=AR(:,13)+rand(A,1)*0.3
figure
plotchar(noisyCharM)
```

BP 网络训练采样全训练样本集，使用所有 26 个大写字母，测试样本采样包含噪声的字母 M 数据点。字母 M 和对应包含噪声的字母 M 图形如图 7-18 所示。

创建 BP 神经网络，并使用全训练数据点训练 BP 神经网络：

```
%创建 BP 神经网络，并使用数据点训练网络
P=AR;
T=TS;
%输入层包含 10 个神经元，输出层为 C2 个神经元，输入/输出层分别使用 logsig 传递函数
net=newff(minmax(P),[10,C2],{'logsig' 'logsig'},'traingdx');
net.trainParam.show=50;
net.trainParam.lr=0.1;
net.trainParam.lr_inc=1.05;
net.trainParam.epochs=3000;
net.trainParam.goal=0.01;
[net,tr]=train(net,P,T);
```

回代检验和测试样本点的检验：

```
%回代检验
A=sim(net,CM);
%测试样本点的检验
a=sim(net,noisyCharM);
%找到字母所在位置
pos=find(compet(a)==1)
figure
%绘制去除噪声后的字母
plotchar(AR(:,pos))
```

包含噪声的字母 M 经过 BP 网络后输出结果如图 7-19 所示。BP 网络去除了字母 M 上的随机噪声。



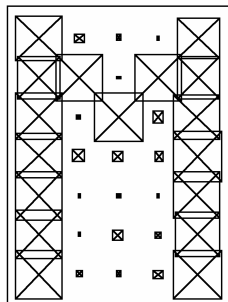


图 7-18 包含噪声的字母 M 图示

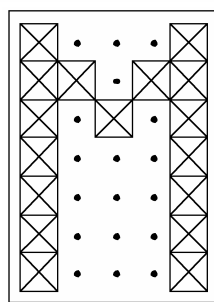


图 7-19 包含噪声的字母 M 经过 BP 网络后输出结果

7.5 径向基网络

径向基函数网络简称径向基网络，即 RBF 网络由 3 层构成，输入层、隐藏层和输出层。同 BP 网络不同的是：径向基函数网络的隐藏层采用径向基 radbas 函数，只有当输入信号靠近 radbas 函数中央时，隐藏层节点才产生较大的输出，因此径向基函数网络具有局部逼近能力。同 BP 网络一样，RBF 网络可以近似逼近任何连续非线性函数。

7.5.1 径向基网络原理

在如图 7-20 所示的径向基神经元模型中， R 为输入向量的个数， w_i 为输入向量的权重系数， b_i 是阈值。

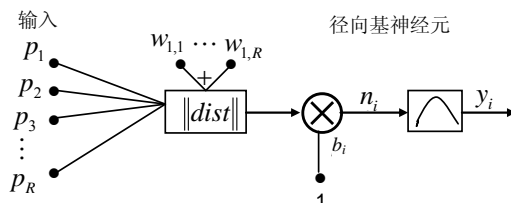


图 7-20 径向基神经元模型

径向基网络中包含多个径向基神经元模型，需要将多个径向基神经元串联，同时有 n 个神经元输出，那么第 i 个径向基神经元的输出为

$$n_i = \|w_{ik} - x_k\| \cdot b_i$$

径向基神经元的输出函数使用径向基函数 radbas，只有当输入靠近 radbas 函数中央时，才会输出较大的值。第 i 个神经元经过径向基传递函数后的输出为

$$y_i = f(n_i) = \text{radbas}(\|W - x\| \cdot b_i)$$

在 MATLAB 命令行中输入以下程序段：

```
%绘制单个径向基函数曲线
x=-4:0.01:4;
y=radbas(x);
subplot(211);
```

```

plot(x,y);
title('radbas 函数曲线')
%绘制多个径向基函数按一定权重合成的曲线
subplot(212);
y2=radbas(x-1);
y3=radbas(x+1.5);
y4=y*1.3+y2*0.7+y3*0.5;
plot(x,y,'b-',x,y2,'r--',x,y3,'g--',x,y4,'k-')
legend('r=0','r=1','r=1.5','合成曲线');

```

运行程序，效果如图 7-21 所示。

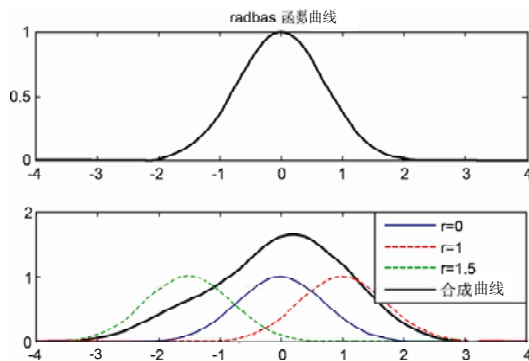


图 7-21 径向基函数曲线

7.5.2 径向基网络相关函数

1. newrb 函数

功能：创建径向基网络。

格式：[net, tr]=newrb(P, T, goal, spread, MN, DF)

说明：其中，P、T 分别为训练样本输入目标输出，goal 为径向基网络输出的总平均误差方差，spread 为径向基函数 radbas 的密度常数，MN 为最大的神经数目。在径向基网络设计中，spread 参数对径向基网络的性能影响较大，通常情况下，spread 值越大时，径向基网络逼近曲线越光滑，当 spread 值过小时，径向基网络的逼近效果就会变差。

在 MATLAB 命令行窗口中输入：

```

P=[1 2 3 4 5];
T=[0.4 1.3 2.1 3.0 4.7];
goal=0.015;
spread=1.5; %径向基函数密度常数
net=newrb(P,T,goal,spread); %创建径向基网络
a=sim(net,P) %回代检验
postreg(a,T) %绘制回归曲线

```

径向基网络输出和训练样本目标输出回归曲线如图 7-22 所示。回代检验的输出结果为：

```

a =
    0.4000    1.3000    2.1000    3.0000    4.7000

```

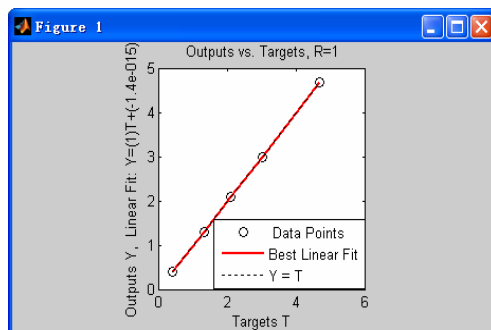


图 7-22 径向基网络输出和训练样本目标输出回归曲线

对新样本进行训练，输入以下代码：

```
>> p=4.2;  
A=sim(net,p)
```

输出结果为：

```
A = 3.3131
```

2. newrb函数

功能：创建一个准确的径向基网络，可以实现网络输出与目标输出零误差。

格式：net=newrb(P, T, spread)

说明：newrb 函数各参数的意义同 newrb 完全相同，具体介绍参见 newrb 函数的相关介绍。

在 MATLAB 命令行中输入以下代码：

```
P=[1 2 3 4 5];  
T=[0.4 1.3 2.1 3.0 4.7];  
spread=1.5; %径向基函数密度常数  
net=newrb(P,T,spread); %创建径向基网络  
a=sim(net,P) %回代检验  
postreg(a,T) %绘制回归曲线  
p=4.2;  
A=sim(net,p)
```

该程序段首先使用 newrb 函数创建一个准确的径向基网络，然后使用训练样本数据回代检验，并绘制网络输出的回归曲线，最后给出测试样本的输出。newrb 函数准确径向基网络输出同目标向量的回归曲线如图 7-23 所示。可以发现此时 newrb 函数创建的准确径向基网络能够实现目标向量零误差输出。回代检验结果和测试样本输出结果如下：

```
a =  
    0.4000    1.3000    2.1000    3.0000    4.7000  
A =    3.3278
```

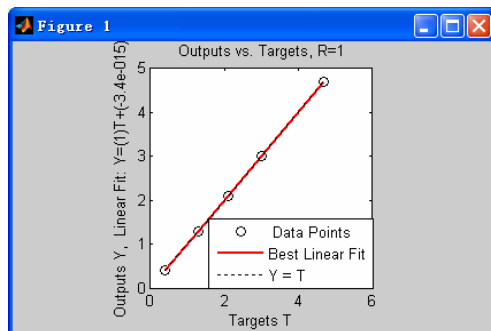
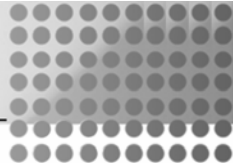


图 7-23 newrb 函数准确径向基网络输出同目标向量的回归曲线



3. newpnn函数

功能：创建概率神经网络。

格式：net=newpnn(P,T,spread)

说明：newpnn 函数参数含义参见 newrb 函数相关介绍，下面通过示例介绍 newpnn 函数的使用。

```
>> P=[0 0 2 3 2 2 1;0 1 2 3 0 1 3];
T=[2 2 1 1 2 1 1];
Tc=ind2vec(T)
```

ind2vec 函数将下标变换成单值向量：

```
Tc =
(2,1)      1
(2,2)      1
(1,3)      1
(1,4)      1
(2,5)      1
(1,6)      1
(1,7)      1
```

创建概率神经网络，并对输入样本进行训练。

```
>> net=newpnn(P,Tc);    %创建概率神经网络
ac=sim(net,P);          %回代检验
a=vec2ind(ac)            %将单值向量转化为下标形式
```

输出结果为：

```
a =
     2     2     1     1     2     1     1
```

从输出结果可以看出，回代误差为 0，对输入向量进行了正确的分类，对新的样本进行训练。

```
>> p=[3 2;1 2];
A=vec2ind(sim(net,p))
```

返回结果为：

```
A =
     1     1
```

4. newgrnn函数

功能：创建广义回归神经网络。

格式：net=newgrnn(P, T, spread)

说明：newgrnn 函数中输入/输出参数同 newrb 完全相同，具体含义参见 newrb 函数介绍。

使用 newgrnn 广义回归神经网络进行逼近。首先创建网络训练数据点：

```
x=-2:0.01:1;
y=2*x.^6+3*x.^5-3*x.^3+x.^2+1;
P=x(1:15:end);
T=y(1:15:end);
```

P、T 为广义回归神经网络的训练样本数据点。以下程序段实现不同的 spread 值下的广义回归神经网络函数逼近。



```

spread=[0.05 0.2 0.4 0.6 0.8];           %6 组不同的 spread 值
l_style={'r-','bo--','ko-','k*--','r^-' , 'bx-'}; %6 组不同的线段形式
for i=1:length(spread)
    net=newgrnn(P,T,spread(i));           %创建广义回归神经网络
    a=sim(net,P);                         %训练样本点回代检验
    plot(P,a,l_style{i})                  %绘制逼近曲线
    hold on;
end
plot(P,T,'o');
legend('spread=0.05','spread=0.2','spread=0.4','spread=0.6','spread=0.8','train data');

```

运行程序, 结果如图 7-24 所示, 从图中可以看出不同的 spread 值下的广义回归神经网络曲线逼近效果差异较大, 当 spread=0.05 时, 曲线逼近效果最好, 与原曲线基本重合, 而随着 spread 数值的增大, 逼近曲线误差增大。

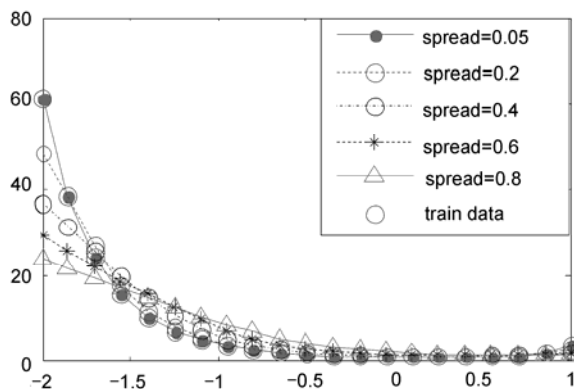


图 7-24 不同的 spread 值下的广义回归神经网络曲线逼近效果

7.5.3 径向基网络应用示例

径向基网络是具有单隐藏层的前向传输网络, 输出函数采用径向基传输函数 `radbas`, 具有非常高的函数逼近精度。

【例 7-6】使用径向基网络进行非线性曲线逼近。

采用非线性曲线, 同例 7-5 中完全相同, 生成训练网络的训练样本数据, 其程序段如例 7-5 所示。

方法一: 使用径向基网络函数 `newrb` 进行非线性曲线逼近代码如下。

```

x=-2:0.01:1;
y=2*x.^6+3*x.^5-3*x.^3+x.^2+1;
P=x(1:15:end);
T=y(1:15:end);
plot(x,y,P,T,'mo');
legend('逼近曲线','网络训练点');
eg=0.02; %总平方误差和
sc=1; %径向基分步密度常数
net=newrb(P,T,eg,sc);
a=sim(net,x);

```

```
figure
plot(P,T,'+',x,a);
legend('网络训练点','径向基网络逼近曲线')
```

运行程序，效果如图 7-25 所示，从图中可以看出，径向基网络曲线逼近效果非常理想。

方法二：使用广义回归神经网络 GRNN 进行非线性曲线逼近。

广义回归神经网络 GRNN 网络训练速度快，非线性映射能力强，同径向基网络相似，特别适合函数逼近。

```
x=-2:0.01:1;
y=2*x.^6+3*x.^5-3*x.^3+x.^2+1;
P=x(1:15:end);
T=y(1:15:end);
plot(x,y,P,T,'mo');
legend('逼近曲线','网络训练点');
spread=0.1;
net=newgrnn(P,T,spread);
a=sim(net,x);
plot(P,T,'+',x,a);
legend('网络训练点','GRNN 网络逼近曲线')
```

运行程序，效果如图 7-26 所示。

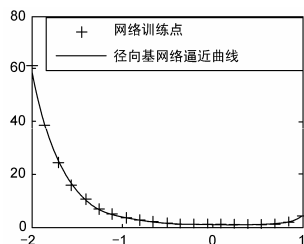


图 7-25 径向基网络曲线逼近效果

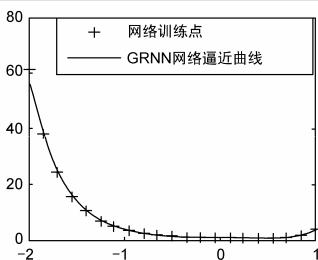


图 7-26 GRNN 网络曲线逼近效果

7.6 回归网络

回归网络（Recurrent Network）也称为记忆网络，主要包括 Hopfield 网络和 Elman 网络。Hopfield 网络通常存储一个或者多个稳定目标向量，当给定一组输入向量时，Hopfield 网络就可以从存储目标向量中找到与输入向量最接近的稳定平衡点输出。

Elman 网络是一个具有两层结构的反向传输网络，其中隐藏层节点的输出直接反馈至隐藏层节点的输入端，因此具有记忆的功能。本节主要介绍 Hopfield 网络和 Elman 网络相关函数及其应用。

7.6.1 回归网络相关函数

1. newhop函数

功能：创建 Hopfield 网络。

格式：net=newhop(T)

说明：其中输入参数 T 为 Q 个目标向量组成的 R×Q 的矩阵，其数值必须为 1 或者 -1。以下代码创建 Hopfield 网络，并仿真网络。

```
T=[-1 -1 1;1 -1 1]; %目标稳定平衡点
net=newhop(T); %创建 Hopfield 网络
ai={[-0.9;-0.8;0.7]}; %测试样本点
[Y,Pf,Af]=sim(net,{1 5},{},ai);
```


运行程序可以得到测试样本点的输出为：

```
>> Y{1}
ans =
    -1
    -1
     1
```

可以看出，测试样本点经过 Hopfield 网络后输出存储的最近的稳定平衡点。

2. newelm函数

功能：创建 Elman 网络。

格式：net=newelm(PR, [S1 S2 ... SN], {TF1 TF2 ... TFN}, BTF, BLF, PF)

说明：由于 Elman 网络结构与 BP 网络基本相同，同样为反向传输网络。

其函数参数说明如下：PR 为 R 维输入元素的 R×2 最大值和最小值矩阵；SN 为第 N 层网络神经元的个数，共有 N 层；TFN 为第 N 层网络的转换函数，默认为 tansig 函数；BTF 为神经网络的训练函数，默认为 trainlm 函数；BLF 为神经网络权值/偏差的学习函数，默认为 learnqdm 函数；PF 为性能评价函数，默认为 mse 函数。Elman 网络训练函数同 BP 网络完全相同。

以下代码创建 Elman 网络并仿真网络。

```
p=round(rand(1,8)); %生成 1×8 的随机数向量
Pseq=con2seq(p); %转化为序列向量
net=newelm([0 1],[5 1],{'tansig','logsig'},'trainlm'); %生成 Elman 网络
a=sim(net,Pseq)
b=seq2con(a);
b{1,1}
```

输出结果为：

```
a =
    [0.5408]    [0.9872]    [0.8403]    [0.9212]    [0.9092]    [0.9109]    [0.9018]    [0.9049]
b{1,1} =
    0.5408    0.9872    0.8403    0.9212    0.9092    0.9109    0.9018    0.9049
```

7.6.2 回归网络的MATLAB实现

【例 7-7】使用 newhop 函数创建 Hopfield 网络，并使用随机数进行训练仿真。

(1) 绘制Hopfield网络状态空间

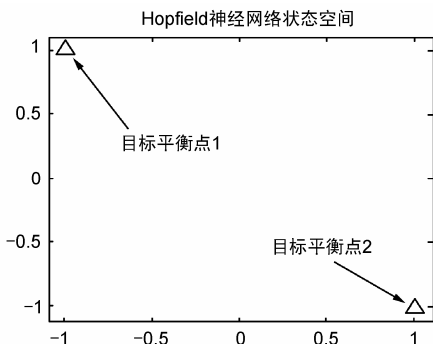


图 7-27 Hopfield 网络状态空间

%定义 Hopfield 网络的目标平衡点

```
T=[1 -1;-1 1]';
plot(T(1,:),T(2,:),'r^');
axis([-1.1 1.1 -1.1 1.1]);
title('Hopfield 网络状态空间');
```

生成的 Hopfield 网络状态空间如图 7-27 所示。

(2) 创建Hopfield网络

```
net=newhop(T); %创建 Hopfield 网络
w=net.LW{1,1} %获取 Hopfield 网络的权重
b=net.b{1,1} %获取 Hopfield 网络的阈值
```

输出结果为:

```
w =
    0.6925   -0.4694
   -0.4694    0.6925
b =
     0
     0
```

(3) 使用原始平衡点仿真网络

```
>> [Y,Pf,Af]=sim(net,2,[],T);
Y
```

返回结果为:

```
Y =
     1     -1
    -1      1
```

返回结果表明, Hopfield 网络确定存储了平衡点数据。

(4) 使用随机数据仿真网络

Hopfield 网络能够寻找到与随机数最相似的稳定平衡点, 以下代码用来测试 Hopfield 网络对随机数的仿真情况:

```
color={'r','g','b','m','y','k'}; %线段颜色
for i=1:6*length(color)
    a={rands(2,1)}; %生成随机数
    [Y,Pf,Af]=sim(net,{1 20},{},a); %Hopfield 网络训练
    re=[cell2mat(a) cell2mat(Y)]; %记录训练过程数据点
    %绘制随机数和稳定平衡点路径
    plot(re(1,1),re(2,1),'rx',re(1,:),re(2,:),color{rem(i,length(color))+1});
    hold on;
end
```

不同随机数 Hopfield 网络输出稳定平衡点的路径如图 7-28 所示。从图中可以看出, 不同的随机数, 经过 Hopfield 网络后, 都将输出与之最近的稳定平衡点。

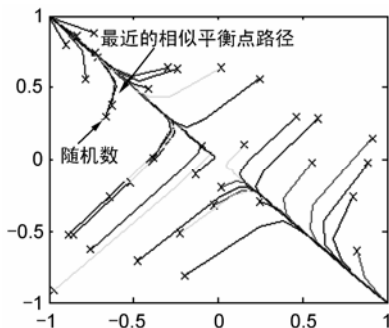


图 7-28 不同随机数 Hopfield 网络输出稳定平衡点的路径

【例 7-8】使用 newelm 函数创建 Elman 网络, 并训练仿真网络。

(1) 输入训练样本数据

```
P=round(rand(1,10)); %输入训练样本
T=[0 0 1 1 0 0 1 1 1]; %训练样本目标输出
```

```
Pseq=con2seq(P)      %转化为序列向量
Tseq=con2seq(T)
```

得到训练样本数据为:

```
Pseq =
    [1]    [0]    [1]    [1]    [1]    [1]    [1]    [0]    [0]    [0]
Tseq =
    [0]    [0]    [1]    [1]    [0]    [0]    [0]    [1]    [1]    [1]
```

(2) 创建Elman网络, 并训练网络

```
%创建两层 Elman 网络, 训练函数为 trainlm
net=newelm([0 1],[5 1],{'tansig','logsig'},'trainlm');
%训练参数设置
net.trainParam.goal=1e-7;
net.trainParam.epochs=499;
net.trainParam.show=49;
net=train(net,Pseq,Tseq); %训练 Elman 网络
```

(3) 回代检验

```
y=sim(net,Pseq);
z=seq2con(y);
```

得到 Elman 网络的输出结果为:

```
>> z{1,1}
ans =
0.0002  0.0000  1.0000  1.0000  0.0003  0.0001  0.0001  0.9992  1.0000  1.0000
```

从结果中可以看出, Elman 网络输出的误差非常小, 具有较好的性能。

第 8 章 MATLAB在自动控制中的应用

在 MATLAB 的控制系统工具箱（Control System Toolbox）中提供了许多仿真函数与模块，用于对控制系统的仿真和分析。

8.1 控制系统模型

8.1.1 控制系统的描述与LTI对象

从数学描述角度，自动控制系统可分为线性系统和非线性系统。由于非线性系统领域太宽，有无数不同的数学描述，也没有统一的通用解法，所以在 MATLAB 中，着重于线性系统的算法。这是因为，在实际应用中对于非线性系统往往可利用小偏差线性化的方法，把某些非线性系统近似为线性系统来求解。在线性系统中，又着重于线性时不变（LTI）系统，或称为常线性系统。

为了分析系统的特性，用户可以选择不同形式的系统模型来描述系统。然而，从系统模型表达式可以看出，无论采取状态空间模型、传递函数模型还是零极点增益模型进行描述，每种方法都需要几个参数矩阵，这对系统的调用和计算都很不方便。根据软件工程中面向对象的思想，MATLAB 通过建立专用的数据结构类型，把线性时不变系统（LTI）的各种模型封装成为统一的 LTI 对象，这样，在一个名称之下包含了该系统的全部属性，大大方便了系统的描述和运算。

MATLAB 控制系统工具箱中规定的 LTI 对象，包含以下三种子对象：ss 对象、tf 对象和 zpk 对象，它们分别与状态空间模型、传递函数模型和零极点增益模型相对应。每个对象都具有其属性和方法，通过对象方法可以存取或者设置对象的属性值。在控制系统工具箱中，这三种对象除了具有 LTI 的共同属性（即子对象可以继承父对象的属性）外，还具有各自特有的属性。LTI 的共同属性如表 8-1 所示。

表 8-1 LTI 的共同属性

属性名称	意 义	属性值的变量类型	属性名称	意 义	属性值的变量类型
Ts	采样周期	标量	OutputName	输出变量名	字符串单元矩阵（数组）
Td	输入时延	数组	Notes	说明	
InputName	输入变量名	字符串单元矩阵(数组)	UserData	用户数据	

① 当系统为离散系统时，给出了系统的采样周期 Ts。Ts=0 或默认时表示系统为连续时间系统；Ts=-1 表示系统是离散系统，但它的采样周期未定。

② 输入时延 Td 仅对连续时间系统有效，其值为由每个输入通道的输入时延组成的时延数组，默认时表示无输入时延。

③ 输入变量名 InputName 和输出变量名 OutputName 允许用户定义系统输入/输出的名称，其值为字符串单元数组，分别与输入/输出有相同的维数，可默认。

④ 说明 Notes 和用户数据 UserData 用以存储模型的其他信息，常用于给出描述模型的三种对象的特有属性，如表 8-2 所示。

表 8-2 三种对象的特有属性

对象名称	属性名称	意 义	属性值的变量类型
tf 对象 (传递函数)	den	传递函数分母系数	由行数组组成的单元阵列
	num	传递函数分子系数	由行数组组成的单元阵列
	variable	传递函数变量	s, z, p, k, z^{-1} 中之一
zpk 对象 (零极点增益)	k	增益	二维矩阵
	p	极点	由行数组组成的单元阵列
	variable	零极点增益模型变量	s, z, p, k, z^{-1} 中之一
	z	零点	由行数组组成的单元阵列
ss 对象 (状态空间)	a	系数矩阵	二维矩阵
	b	系数矩阵	二维矩阵
	c	系数矩阵	二维矩阵
	d	系数矩阵	二维矩阵
	e	系数矩阵	二维矩阵
	StateName	状态变量名	字符串单元向量

8.1.2 典型系统的生成

在 MATLAB 控制系统中, 提供一些常见的线性时不变 (LTI) 系统的生成函数。

1. rss 函数

功能: 随机生成 N 阶稳定的连续状态空间模型。

格式: sys=rss(N, P, M)

说明: 该系统具有 M 个输入, P 个输出。默认时 P=M=1, 即 sys=rss(N)。

【例 8-1】 利用 rss 函数生成三阶稳定的连续状态空间系统。

```
>> sys=rss(3)
```

输出:

```
a =
      x1      x2      x3
x1  -0.7694 -0.03199  0.4178
x2  -0.03199 -0.5946 -0.1948
x3   0.4178 -0.1948 -0.6074
b =
      u1
x1    -0
x2  2.183
x3    -0
c =
      x1      x2      x3
y1     0  1.067     0
d =
      u1
y1 -0.09565
Continuous-time model.
```

2. rmodel函数

功能：随机生成 M 阶稳定的连续线性模型系数。

格式：[num, den]=rmodel(N, P)

[A, B, C, D]=rmodel(N, P, M)

说明：生成一个 N 阶连续的状态空间模型系统，具有 M 个输入系统， P 个输出系统。函数 rmodel 仅用于产生 LTI 对象的系数，它并不生成 LTI 对象本身。

【例 8-2】利用 rmodel 函数生成三阶连续的传递函数模型系统。

```
>> [num,den]=rmodel(3)
```

得到传递函数模型的系数为：

```
num =
    0.8580   -1.4124   -0.3926    0.6884
den =
    1.0000    2.0402    1.0511    0.1535
```

3. drss和drmodel函数

drss 和 drmodel 函数的用法与 rss 和 rmodel 函数的用法相似，不同点仅仅在于它生成的是离散系统。

【例 8-3】生成一个三阶两输入两输出的稳定离散状态空间系统。

```
>> sys=drss(3,2,2)
```

得到为：

```
a =
      x1      x2      x3
x1    0.2398    0.3824   -0.5384
x2    0.007056   -0.21    -0.387
x3   -0.6603   -0.0598    0.405
b =
      u1      u2
x1    1.254    0.5711
x2   -1.594     -0
x3   -1.441    0.69
c =
      x1      x2      x3
y1    0.8156     0    1.191
y2    0.7119    0.6686   -1.202
d =
      u1      u2
y1     -0     -0
y2     -0    0.2573
Sampling time: unspecified
Discrete-time model.
```

4. ord2 函数

功能：生成固有频率为 W_n ，阻尼系数为 Z 的连续二阶状态空间模型系统。

格式：[A, B, C, D]=ord2(Wn, Z)

[num, den]=ord2(Wn, Z)

说明：该函数也用来产生二阶系统的系数，不能生成系统本身，因此，它的左端输出变量的数目为四个或两个，决定了生成的系统属于状态空间还是传递函数类型。

【例 8-4】生成一个具有如下传递函数的连续二阶系统的传递函数模型和状态空间模型，其中 $\omega_n = 10$ ， $\xi = 0.5$ 。

$$H(s) = \frac{1}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

```
>> [num,den]=ord2(10,0.5)
num =      1
den =      1     10     100
>> [A,B,C,D]=ord2(10,0.5)
A =      0      1
    -100    -10
B =      0
      1
C =      1      0
D =      0
```

5. pade函数

格式：sysx=pade(sys, N)

功能：对连续系统 sys 产生 N 阶 pade 近似的延迟后，生成新的系统 sysx。

8.1.3 连续系统与采样系统之间的转换

随着计算机在控制系统中的广泛使用，采样系统的分析设计也变得更加普遍和重要。所谓采样系统是指将连续系统的部分控制部分进行离散化，形成一类由连续部分和采样离散部分混合构成的系统。由于采样系统方程比较容易求解，且所得的结果接近实时运行，因此，人们往往把连续系统有意地转化为性能相当的采样系统；反过来，有时人们用测量和辨识的方法，得到系统差分方程模型，希望由它求得相应实际物理世界的连续系统模型。

从连续系统到采样系统的转化关系如下。若连续系统的状态方程为

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}$$

则对应的采样系统状态方程为

$$\begin{aligned}x(k+1) &= A_d x(k) + B_d u(k) \\ y(k) &= C_d x(k) + D_d u(k)\end{aligned}$$

式中， $A_d = e^{At}$ ， $B_d = \int_0^{T_s} e^{A(t-\tau)} B d\tau$ ， $C_d = C$ ， $D_d = D$ ， T_s 为采样周期。

反之，从采样系统到连续系统的转化关系为上式的逆过程

$$A = \frac{1}{T_s} \ln A_d, \quad B = (A_d - I)^{-1} A_d B_d, \quad C = C_d, \quad D = D_d$$

需要指出的是，虽然算式简明，但因为这些系数都是矩阵，连续系统与采样系统之间的转换计算是十分繁杂的，即使是三阶系统，用手工进行运算也是非常困难的。因此，计算机辅助设计在这个领域就更显得不可缺少。在 MATLAB 控制工具箱中提供了相关函数命令。

1. c2d函数

格式: `sysd=c2d(sysc, Ts, method)`

`[sysd,G]=c2d(sysc, Ts, method)`

功能: 把连续系统 `sysc` 按指定的采样周期 `Ts` 和 `method` 方法, 转换为采样系统 `sysd`。其中 `method` 共有五种选择, 对应下列字符串。对于状态空间模型, 将连续系统的初始条件映射成离散初始条件存放于矩阵 `G` 中, 如系统 `sysc` 的初始条件为 `x0`、`u0`, 则离散系统的初始条件为 `xd[0]=G*[x0; u0]`, `ud[0]=u0`。

'zoh': 零阶保持器 (默认值)。

'foh': 一阶保持器。

'tustin': 双线性变换 (`tnsth`) 法。

'prewarp': 频率预修正双线性变换法, 用此法时还增加一个变元 (边缘频率 `Wc`), 即调用格式为 `sysd=c2d(sysc, Ts, 'prewarp', Wc)`。

'matched': 根匹配法。

2. d2d函数

格式: `sys=d2d(sys, Ts)`

功能: 将采样系统 `sys` 按采样周期 `Ts` 重新采样形成一个等效的离散系统。转换过程是, 先将待变换的采样系统按零阶保持器转换为原来的连续系统, 然后再用新的采样频率和零阶保持器转换为新的采样系统。

【例 8-5】系统的传递函数为

$$H(s) = \frac{1.5s^2 + 3s + 1}{s^2 + 2s + 2}$$

输入延时 $T_d = 0.3s$, 试用一阶保持法对连续系统进行离散, 采样周期为 $T_s = 0.1s$ 。

MATLAB 代码为:

```
%利用 tf 函数生成连续系统的传递函数模型 (tf 用法参见此例)
sys=tf([1.5 3 1],[1 2 2],'td',0.3);
>> sysd=c2d(sys,0.1,'foh'); %形成采样系统
```

程序运行结果为:

```
Transfer function:
      1.497 z^2 - 2.713 z + 1.225
z^(-3) * -----
      z^2 - 1.801 z + 0.8187
Sampling time: 0.1
```

8.2 控制系统的时域分析

8.2.1 时域分析的一般方法

一个动态系统的特性经常用典型输入下的时间响应来描述。所谓响应是指零初值条件下某种典型输入函数作用下控制对象的响应, 控制系统中常用的输入函数为单位阶跃函数和脉冲激励函数 (即冲激函数)。MATLAB 的控制系统工具箱中提供了求取这两种典型输入函数下系统响应的函数。

step函数

功能：该函数用来求取系统的阶跃响应。

格式：[y, x]=step(num, den, t)

[y, x]=step(A, B, C, D, iu, t)

说明：[y, x]=step(num, den, t)函数适用于传递函数表示的系统模型，其中，num 和 den 分别为线性系统传递函数的分子和分母多项式系数，t 为选定的仿真时间，返回值 y 为系统在仿真时刻各个输出所组成的矩阵，而 x 为自动选择的状态变量的时间响应数据。当该函数没有返回值时，MATLAB 将直接在屏幕上绘制系统的阶跃响应曲线。[y, x]=step(A, B, C, D, iu, t)函数适用于由状态方程表示的系统模型，其中，(A, B, C, D) 是系统的状态模型，iu 是输入变量的序号，返回值 y 为系统在仿真时刻各个输出所组成的矩阵，而 x 为自动选择的状态变量的时间响应数据。

【例 8-6】含有零点的二阶系统的传统函数 $G(s) = \frac{\omega_n^2(T_m s + 1)}{s^2 + 2\xi\omega_n s + \omega_n^2}$ ，设其固有频率 $\omega_n = 1$ ，

阻尼系数 $\xi = 0.5$ ，在 T_m 分别为 0.5, 1, 2 时，分别画出其阶跃响应函数。将该系统在条件 $T_s = 0.1$ 下离散化，再求阶跃响应。

代码如下：

```
wn=1;Ts=0.5;z=0.4;
for Tm=[0.5 1 2]
    sys=tf([Tm,1]*wn^2,[1,2*z*wn,wn^2]);    %生成不同的 LTI 模型——连续系统
    sysd=c2d(sys,Ts);                        %连续系统转换成采样系统
    figure(1),step(sys),hold on;              %连续系统的阶跃响应
    figure(2),step(sysd),hold on;             %采样系统的阶跃响应
end
```

运行程序，结果如图 8-1 所示。从图中可以看出，所有的零点越小，时间常数 T_m 越大，阶跃过渡过程的超调加大，上升时间减小，使系统的跟踪速度加快。

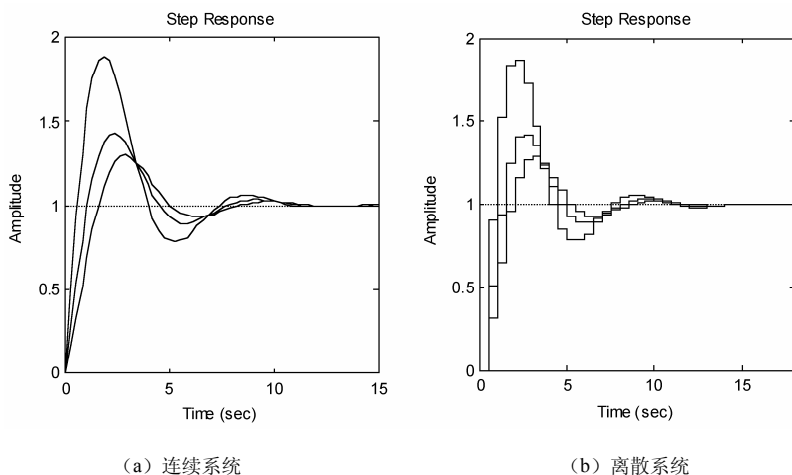


图 8-1 有零点的二阶系统在不同阻尼系数下的阶跃响应曲线

求取脉冲响应函数 impulse() 的调用方法与 step() 函数基本一致。下面通过几个示例进一步讨论控制系统时域分析的一般方法。

【例 8-7】已知某二阶系统为 $G(s) = \frac{\omega_n^2(T_m s + 1)}{s^2 + 2\xi\omega_n s + \omega_n^2}$, $\xi = 0.5$, $\omega_n = 4$, 求其阶跃响应。

代码如下:

```
num=16;
den=[1 4 16];
t=0:0.02:2;
c=step(num,den,t);
plot(t,c);
xlabel('Time-sec');
ylabel('y(t)'); grid on;
title('Two order linear system');
```

首先说明一下时间 t 的选择方法。对于二阶系统, 其过渡时间的近似公式为

$$t_s = \frac{2 \sim 3}{\xi\omega_n} \quad (8-1)$$

由式 (8-1), 可以求出此系统输出 y 的过渡时间在 $1 \sim 1.5s$, 即系统经过这个时间后逐渐趋于稳态, 所以将仿真时间 t 的取值范围定在 $0 \sim 2s$ 之间, 共输出 100 个点, 因此选择时间间隔为 $0.02s$ 。程序运行后, 二阶系统的阶跃响应曲线如图 8-2 所示。

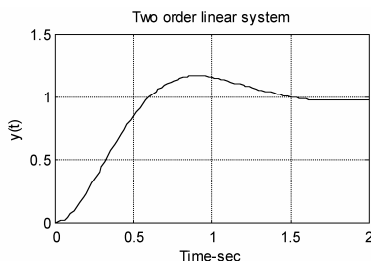


图 8-2 二阶系统的阶跃响应曲线

【例 8-8】求多输入/输出系统的单位阶跃响应和单位冲激响应。

$$x' = \begin{bmatrix} 2.25 & -5 & -1.25 & -0.5 \\ 2.25 & -4.25 & -1.25 & -0.25 \\ 0.25 & -0.5 & -1.25 & -1 \\ 1.25 & -1.75 & -0.25 & -0.75 \end{bmatrix} x + \begin{bmatrix} 4 & 6 \\ 2 & 4 \\ 2 & 2 \\ 0 & 2 \end{bmatrix} u, \quad y = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \end{bmatrix} x$$

代码如下:

```
a=[2.25 -5 -1.25 -0.5;2.25 -4.25 -1.25 -0.25;
    0.25 -0.5 -1.25 -1;1.25 -1.75 -0.25 -0.75];
b=[4 6;2 4;2 2;0 2];
c=[0 0 0 1;0 2 0 2];
d=zeros(2,2);
figure(1);step(a,b,c,d); %求系统的单位阶跃响应
figure(2);impz(a,b,c,d); %求系统的单位冲激响应
```

运行程序结果如图 8-3 所示。

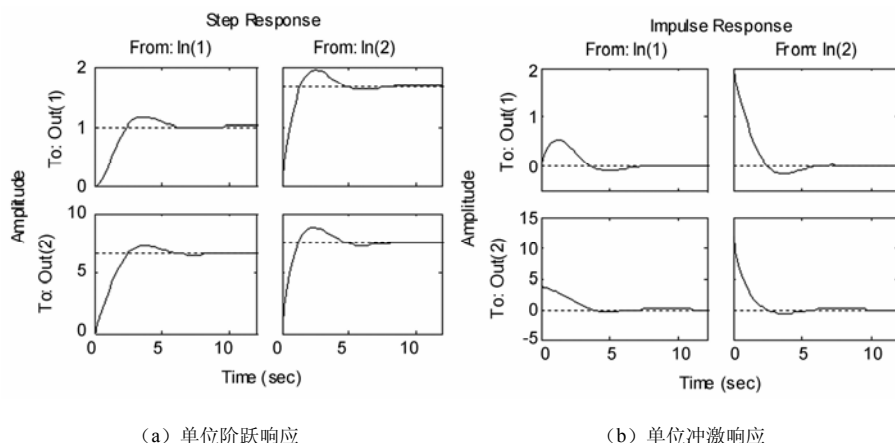


图 8-3 输出图形

8.2.2 常用时域分析函数

对于控制系统而言, 系统的数学模型实际上是某中微分方程或差分方程的模型, 因而在仿真过程中需要用某种数值算法从给定的初始值出发, 逐步计算出每一个时刻系统的响应, 即系统的时间响应, 最后绘制出系统的响应曲线, 由此来分析系统的性能。时间响应探究系统对输入和扰动在时域内的瞬态行为, 系统特征如上升时间、调节时间、超调量和稳态误差都能从时间响应上反映出来。MATLAB 除了提供了前面介绍的对象系统阶跃响应、冲激响应等进行仿真的函数外, 还提供了大量对控制系统进行时域分析的函数, 如表 8-3 所示。

表 8-3 常用时域分析函数

函 数	说 明	函 数	说 明
covar	连续系统对白噪声的方差响应	lsim	连续系统对任意输入的响应
dcovar	离散系统对白噪声的方差响应	dlsim	离散系统对任意输入的响应
impz	连续系统的脉冲响应	step	连续系统的阶跃响应
dimpz	离散系统的脉冲响应	dstep	离散系统的阶跃响应
initial	连续系统的零输入响应	filter	数字滤波器
dinitial	离散系统的零输入响应		

下面示例对部分函数进行了说明, 读者可以从中了解到这些函数的使用方法。

(1) initial函数

【例 8-9】某三阶系统为
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0.5 \\ 2 & -2 & 0.3 \\ 1 & -4 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \quad y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

当初时状态 $x_0 = [0 \ 0 \ 1]^T$ 时, 求该系统的零输入响应。

代码如下:

```
a=[1 -1 0.5;2 -2 0.3;1 -4 -0.1];
b=[0 0 1]';
c=[0 0 1];
```

```

d=0;
x0=[1 0 0]';
t=0:0.01:20;
initial(a,b,c,d,x0,t);
title('The initial condition response');

```

运行程序，效果如图 8-4 所示。

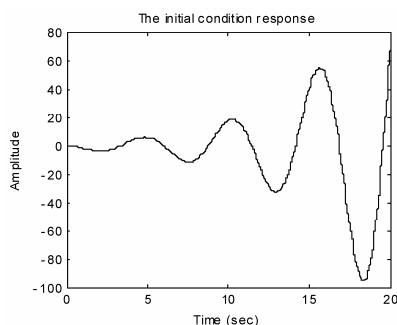


图 8-4 连续系统的零输入响应

(2) dinitial 函数

【例 8-10】某离散二阶系统为

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} -0.6 & -0.3162 \\ 0.3162 & 0 \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$y = [2.4 \quad 6.0083] \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + u$$

当系统的初始状态为 $x_0 = [1 \ 0]^T$ 时，求系统的零输入响应。

代码如下：

```

a=[-0.6 -0.3162;0.3162 0];
b=[1 0]';
c=[2.4 6.0083];
d=1;
x0=[1 0]';
dinitial(a,b,c,d,x0);

```

运行程序，效果如图 8-5 所示。

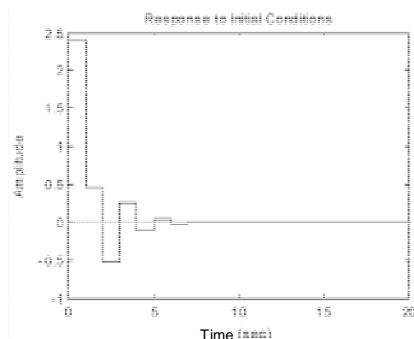


图 8-5 离散系统的零输入响应

(3) lsim函数

【例 8-11】已知某系统为 $H(s) = \frac{s^3 + 6.8s^2 + 13.85s + 8.05}{s^5 + 11.2s^4 + 46.4s^3 + 88.4s^2 + 77.4s + 25.2}$ ，求周期为 6s 的方波输出响应。

代码如下：

```
num=[1.0000 6.8000 13.8500 8.0500];
den=[1.0000 11.2000 46.4000 88.4000 77.4000 25.2000];
t=0:0.1:15;
%构造周期为 6 的方波
period=6;
u=(rem(t,period)>=period./2);
lsim(num,den,u,t);
```

运行程序，效果如图 8-6 所示。

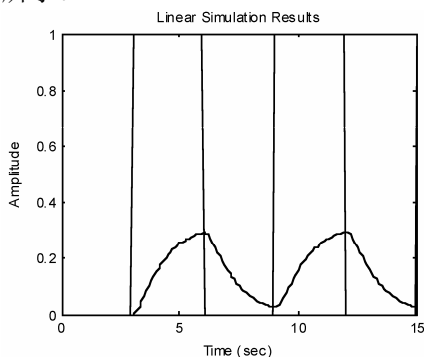


图 8-6 连续系统的输出响应

(4) dlsim函数

【例 8-12】某二阶系统为 $H(s) = \frac{2s^2 - 3.4s + 1.5}{s^2 - 1.6s + 0.8}$ ，求出系统对 50 点随机噪声的响应曲线。

代码如下：

```
num=[2 -3.4 1.5];
den=[1 -1.6 0.8];
u=rand(50,1);
dlsim(num,den,u);
```

运行程序，效果如图 8-7 所示。

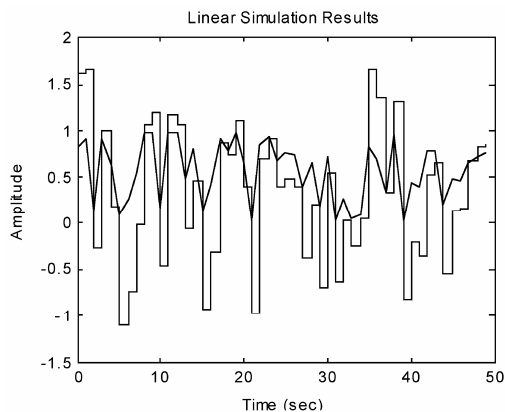


图 8-7 离散系统的输出响应

8.2.3 时域分析应用示例

本节在前面介绍的基础上，通过综合性的示例进一步讨论控制系统的时域分析。

【例 8-13】典型二阶系统为 $G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$ ，其中 ω_n 为自然频率（无阻尼振荡频率）；

ξ 为阻尼系数。要求绘制出当 $\omega_n = 4$ ， ξ 分别为 0.1, 0.2...1.0, 2.0 时，系统的单位阶跃响应。

代码如下：

```
wn=4;
k=[0.1:0.1:1,2];
figure(1)
hold on;
for i=k
    num=wn.^2;
    den=[1,2*i*wn,wn.^2];
    step(num,den);
end
title('The step response of two order system');
```

运行程序，效果如图 8-8 所示。

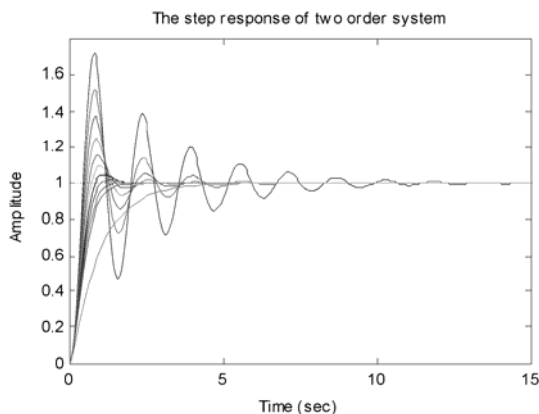


图 8-8 典型二阶系统的单位阶跃响应曲线

【例 8-14】已知某系统为

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} -1.5 & -0.8 & 0 & 0 \\ 0.8 & 0 & 0 & 0 \\ 0.3 & 0.4 & -4.0 & -1.25 \\ 0 & 0 & -1.25 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

以 $t = 0.6$ 为取样周期，将系统转换为离散系统，然后求出离散系统的单位阶跃响应、冲激响应与零输入响应（ $x_0 = [1 \ 1 \ 1 \ 1]^T$ ）。

代码如下：

```
a1=[-1.5 -0.8 0 0;0.8 0 0 0;0.3 0.4 -4.0 -1.25;0 0 -1.25 0];
b1=[1 0 1 0]';
c1=[1 2 1 2];
d1=0;
t=0.6;
[a,b,c,d]=c2dm(a1,b1,c1,d1,t,'tustin');
subplot(221);
dstep(a,b,c,d);
subplot(222);
dimpulse(a,b,c,d);
subplot(223);
x0=[1 1 1 1];
dinitial(a,b,c,d,x0);
axis([0 6 -0.5 2.5]);
subplot(224);
[z,p,k]=ss2zp(a,b,c,d,1);
zplane(z,p);
title('Discrete pole-zero map');
```

运行程序，效果如图 8-9 所示。

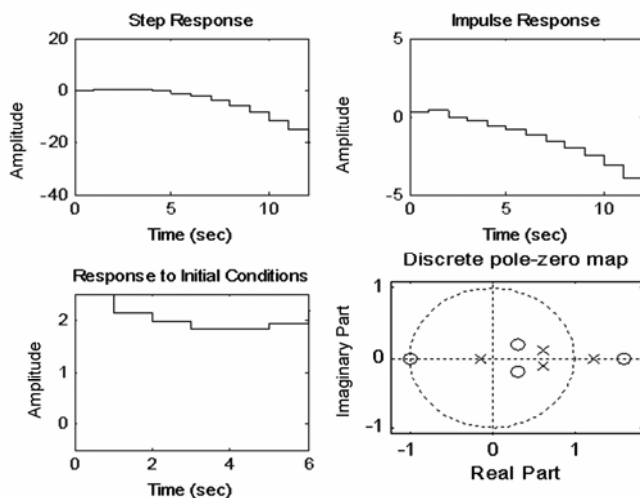


图 8-9 离散系统的响应与零极点图

8.3 根轨迹分析

根轨迹法是 W.R.Evans 在 1948 年提出的一种求解闭环特征方程根的图解方法，由于它的计算量最小，而且非常直观化，因而从其一诞生就被广泛地应用于工程实际当中。该方法根据系统开环传递函数的极点和零点分布，依照一些简单的规则，用作图的方法求出闭环极点的分布，避免了复杂的数学运算。

MATLAB 绘制根轨迹的函数为 `rlocus(num, den, K)`，其中，`num` 和 `den` 分别是系统开环传递函数的分子多项式和分母多项式系数，`K` 为开环增益，`K` 的范围可以指定，若不给定 `K` 的范围，则隐含 `K` 从 0 到 $+\infty$ ，该函数可以精确地绘制出根轨迹。

8.3.1 模条件和角条件

设开环传递函数为

$$G_o(s) = \frac{KN(s)}{D(s)} \quad (8-2)$$

式中, $N(s)$ 和 $D(s)$ 分别是 s 的 m 次多项式和 n 次多项式, 且 $m \leq n$, 则系统的闭环特征方程为

$$KN(s) + D(s) = 0 \quad (8-3)$$

显然, $N(s)$ 和 $D(s)$ 没有公因子时, 该方程与下面的方程是同根的。

$$G_o(s) = -1 \quad (8-4)$$

$G_o(s)$ 总可以写成如下形式

$$G_o(s) = \frac{KN(s)}{D(s)} = \frac{K(s-z_1)\cdots(s-z_m)}{(s-p_1)\cdots(s-p_n)} \quad (8-5)$$

式中, $K > 0$, z_1, \cdots, z_m 为系统的开环零点, p_1, \cdots, p_n 为系统的开环极点, $n \geq m$ 。

对式 (8-5) 两边取模, 得

$$|G_o(s)| = K \cdot \frac{\prod_{i=1}^m |s-z_i|}{\prod_{j=1}^n |s-p_j|} \quad (8-6)$$

对式 (8-5) 两边取角, 得

$$\arg G_o(s) = \sum_{i=1}^m \arg(s-z_i) - \sum_{j=1}^n \arg(s-p_j) \quad (8-7)$$

综合式 (8-5), 得根轨迹的模条件和角条件如下所示:

$$K \cdot \frac{\prod_{i=1}^m |s-z_i|}{\prod_{j=1}^n |s-p_j|} = 1 \quad (8-8)$$

$$\sum_{i=1}^m \arg(s-z_i) - \sum_{j=1}^n \arg(s-p_j) = (2k+1)\pi \quad (8-9)$$

8.3.2 绘制根轨迹的规则

绘制根轨迹应满足以下条件

① 根轨迹的对称性: 根轨迹关于实轴对称。

② 根轨迹的分支数, 起点, 终点: 对于 $n \geq m$ 的系统而言, 根轨迹共有 n 条分支。根轨迹起始于开环极点, 终止于有限的和无穷远的开环零点。

③ 根轨迹在实轴上的分布：实轴上的一段属于根轨迹，当且仅当其右侧开环传递函数的实零点和实极点数之和为奇数。

④ 根轨迹的渐近线： $n-m$ 条渐近线都从实轴上的共同交点向外辐射，而辐射角正是渐近线的方向角，满足

$$\theta = \frac{2k+1}{n-m} \pi$$

渐近线与实轴交点的坐标可以表示为

$$\text{渐近线与实轴交点的坐标} = \frac{\sum[\text{极点坐标}] - \sum[\text{零点坐标}]}{\text{极点数} - \text{零点数}}$$

⑤ 根轨迹的分离点与汇合点：如果实轴上两个相邻极点间的线段属于根轨迹，则它们之间必须有分离点。同理，如果实轴上两个相邻零点之间的线段属于根轨迹，则它们之间必有汇合点。确定分离点和汇合点的方法有重根法和试点法，利用角条件求实轴上的分离点和汇合点。

⑥ 实轴上分离点的分离角和汇合点的汇合角：实轴上分离点的分离角恒为 $\pm 90^\circ$ ，同理，实轴上汇合点的汇合角也恒为 $\pm 90^\circ$ 。

⑦ 根轨迹在复极点（或复零点）处的出射角（或入射角）：出射角就是从复极点 p_i 出发的根轨迹切线的方向角 θ ， θ 可以由下式求出

$$\theta = \sum[\text{各零点指向本极点的方向角}] - \sum[\text{其他零点指向本极点的方向角}] + \text{极点的反向角} \quad (8-10)$$

同理，可求出复零点的入射角。

⑧ 根轨迹与虚轴的交点及临界根轨迹增益值：将 $s = j\omega$ 代入特征方程即可求得 ω 和 K ，即根轨迹与虚轴的交点坐标及交点对应的临界根轨迹增益值 K_{cr} 。

常用的根轨迹函数有 pzmap、rlocfind、rlocus、sgrid 及 zgrid，这些函数的相应用法请参看以下示例。

8.3.3 根轨迹的应用示例

本节就结合工程实际，通过示例来讨论 MATLAB 中应用根轨迹法分析控制系统的基本方法。

【例 8-15】设系统的开环传递函数为 $H(s) = \frac{K(s+5)}{s^3 + 5s^2 + 6s}$ ，要求绘制出闭环系统的根轨迹，并确定交点处的增益。

代码如下：

```
num=[1 5];
den=[1 5 6 0];
rlocus(num,den);           %绘制出根轨迹
[k,p]=rlocfind(num,den)    %确定增益及相应的闭环极点
title('Root locus');
gtext('k=0.5');           %用鼠标标示文本
```

执行时先绘制出根轨迹，并提示用户在图形窗口中选择根轨迹上的一点，以计算出增益 K 及相应的极点。这时十字光标放在需要选取的根轨迹的交点处，即可得到如下数据：

```

selected_point =
-2.2761 + 0.0466i
k =
0.1697
p =
-3.0953
-1.7478
-0.1569

```

这说明闭环系统有 3 个极点。事实上, 如果能够将十字光标准确地放在根轨迹的交点之上, 应有 $p_2=p_3$ 。系统的根轨迹如图 8-10 所示。

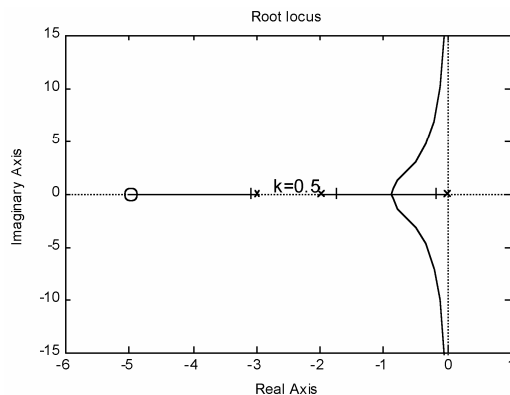


图 8-10 系统的根轨迹

【例 8-16】已知一离散系统的开环传递函数 $H(z) = \frac{2z^2 - 0.5z + 2}{z^2 - 1.8z + 0.9}$, 要求绘制其闭环系统的根轨迹, 并绘制出网络线。

代码如下:

```

num=[2 -0.5 2];
den=[1 -1.8 0.9];
axis('square'); %将图形绘制在正方形区域内
rlocus(num,den);
title('Root locus of discrete system');
zgrid; %绘制网格线

```

运行程序, 效果如图 8-11 所示。

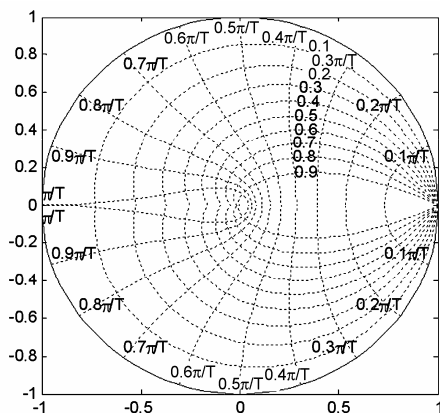


图 8-11 离散系统的根轨迹图

【例 8-17】已知开环传递函数 $H(s) = \frac{K(s+2)}{(s^2+4s+4)^2}$ ，要求绘制该系统的闭环根轨迹，分析其稳定性，并绘制出当 $K=46$ 和 $K=56$ 时系统的闭环冲激响应。

代码如下：

```
num=[1 2];
den1=[1 4 3];
den=conv(den1,den1);
figure(1);
k=0:0.1:150;
rlocus(num,den,k);    %绘制图 8-12
title('Root locus');
[k,p]=rlocfind(num,den)

%检验系统的稳定性
figure(2);
k=46;
num1=k*[1 2];
den=[1 4 3];
den1=conv(den,den);
[num,den]=cloop(num1,den1,-1);
impz(num,den);        %绘制图 8-13
title('Impulse response (k=45)');

%检验系统的稳定性
figure(3);
k=56;
num1=k*[1 2];
den=[1 4 3];
den1=conv(den,den);
[num,den]=cloop(num1,den1,-1);
impz(num,den);        %绘制图 8-14
title('Impulse response (k=56)');
Select a point in the graphics window
```

该程序先绘制出闭环系统的根轨迹图，然后求出系统的临界稳定增益，最后用两组特定的 K 值所对应的系统的闭环冲激响应来验证系统的稳定性。

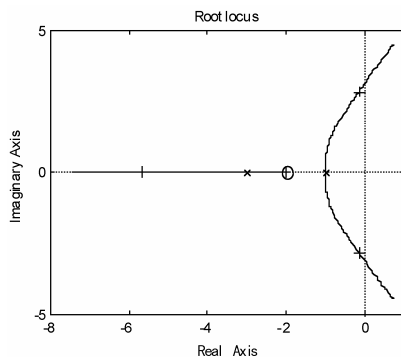


图 8-12 闭环系统的根轨迹图

接着,系统要求在根轨迹上用鼠标输入要标示增益和相应的闭环极点,执行后得到的结果如下:

```
selected point =
-5.9206 - 0.0466i
k =
52.6929
p =
-5.9209
-0.0300 + 3.0932i
-0.0300 - 3.0932i
-2.0190
```

这里利用 `rlocfind` 函数来找出根轨迹与虚轴的交点,并求得交点处 $K = 52.6929$,也就是说,当 $K < 52.6929$ 时,闭环系统稳定, $K > 52.6929$ 时,闭环系统不稳定。这也可以从该系统的闭环冲激响应看出,分别取 $K=45$ 和 $K=56$,绘制出该系统的闭环冲激响应如图 8-13 及图 8-14 所示。

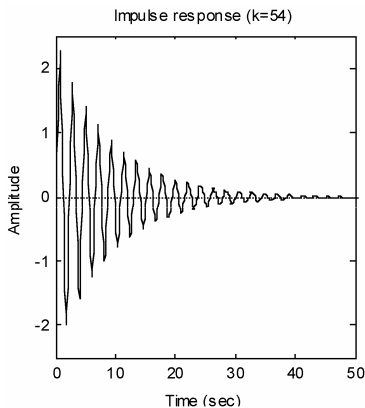


图 8-13 $K=45$ 时系统的闭环冲激响应

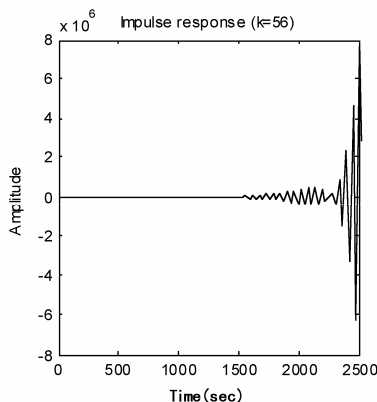


图 8-14 $K=56$ 时系统的闭环冲激响应

从图 8-13 及图 8-14 可以看出,当 $K=45$ 时,闭环系统稳定;当 $K=56$ 时,闭环系统不稳定,其冲激响应是发散的。

8.4 控制系统的频域分析

8.4.1 幅相频率特性

1. Nyquist图

幅相频率特性曲线(Nyquist 曲线)以角频率 ω 为参变量,当 ω 由 0 变为 $+\infty$ 时,系统的频率特性构成的向量 $G(j\omega) = A(\omega)e^{j\varphi(\omega)}$ 的始端在复平面上形成的曲线,称为系统的幅相频率特性曲线,即 Nyquist 曲线。系统的传递函数 $G(s)$ 为

$$G(s) = \frac{\sum_{i=1}^m b_i s^{m-i+1} + b_{m+1}}{\sum_{j=1}^n a_j s^{n-j+1} + a_{n+1}} = \frac{b_1 s^m + b_2 s^{m-1} + \cdots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + \cdots + a_{n+1}}$$

将系统的传递函数 $G(s)$ 转化到复平面下,即 $s = j\omega$, 于是得到

$$G(j\omega) = \frac{b_1(j\omega)^m + b_2(j\omega)^{m-1} + \cdots + b_{m+1}}{a_1(j\omega)^n + a_2(j\omega)^{n-1} + \cdots + a_{n+1}} = P(\omega) + jQ(\omega) = A(\omega)e^{j\varphi(\omega)}$$

其中, $P(\omega)$ 为系统频率特性的实部, $Q(\omega)$ 为系统频率特性的虚部; $A(\omega)$ 为频率特性的模, $\varphi(\omega)$ 为频率特性的幅角。根据复数定义可以知道:

$$A(\omega) = \sqrt{P^2(\omega) + Q^2(\omega)}$$

$$\varphi(\omega) = \arctg(Q(\omega)/P(\omega))$$

当绘制 $A(\omega)$ 和 $\varphi(\omega)$ 随着角频率 ω 的变化情况时, 分别对应于系统的幅频特性和相频特性。那么在复平面中由 $\Omega = \{(A(\omega), \varphi(\omega)) | \omega = 0 \rightarrow \infty\}$ 所构成的所有点的集合就是系统幅相频率特性曲线 (Nyquist 图)。

2. 用开环幅相频率特性判断闭环系统的稳定性

一个闭环系统是否稳定主要取决于该闭环系统的特征方程的所有根是否位于 S 平面的左半平面。当闭环系统特征方程的根位于虚轴上时, 那么系统处于临界稳定。使用开环系统的幅相频率特性来判断闭环系统的稳定性主要有以下两条判据。

① 如果开环系统稳定, 那么闭环系统稳定的条件是: 当角频率 ω 由 $-\infty$ 变为 $+\infty$ 时, 开环幅相频率特性曲线 (Nyquist 图) 不包围 $(-1, j0)$ 这一点。

② 如果开环系统不稳定, 那么闭环系统稳定的条件是: 若开环系统传递函数在右半平面有 p 个极点, 则当角频率 ω 由 0 变为 $+\infty$ 时, 开环频率特性曲线在复平面上逆时针穿过 $(-1, j0)$ 点 $p/2$ 次, 则闭环系统稳定, 否则闭环系统不稳定。

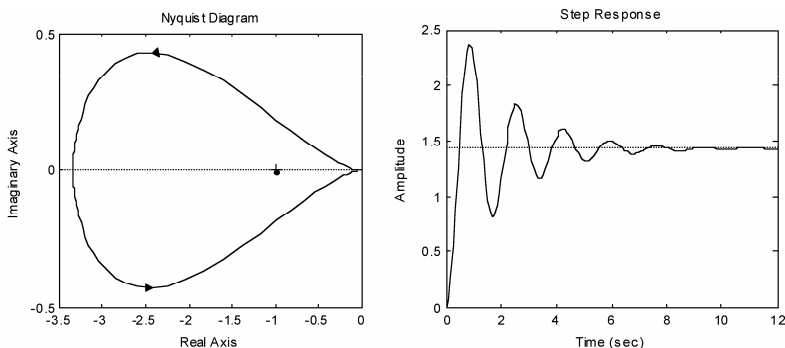
3. MATLAB应用示例

【例 8-18】已知开环系统 $G(s) = \frac{20}{(s-2)(s+3)}$, 绘制系统的 Nyquist 曲线图, 并判断系统的稳定性, 给出闭环系统的阶跃响应曲线。

代码如下:

```
s=zpk([], [2 -3], 20);
nyquist(s);           %绘制开环系统的 Nyquist 曲线图
s2=feedback(s,1);     %控制系统的单位负反馈, 即闭环系统
figure(2)
step(s2);             %绘制闭环系统的阶跃响应曲线
```

运行程序, 效果如图 8-15 所示, 从开环系统 Nyquist 曲线可以看出, 逆时针穿过 0.5 次, 而开环系统包含右平面上一个极点 $p=2$ 。因此可以看出, 闭环系统是稳定的, 图 8-15(b)所示的闭环系统阶跃响应曲线同样表明闭环系统是稳定的。



(a) Nyquist 曲线图

(b) 闭环系统的阶跃响应曲线

图8-15 开环系统的Nyquist曲线和闭环系统的阶跃响应曲线

增加开环系统的零点或极点, 控制系统的稳定性将会发生闭环, 在例 8-18 中的控制系统中加入 $z=0$ 的零点和 $p=1$ 的极点, 判断新系统的稳定性。

```
s=zpk([0],[1 2 -3],20);    %零极点模型
nyquist(s);
s2=feedback(s,1);          %单位负反馈
figure(2)
step(s2);                  %反馈系统的阶跃响应
```

运行程序, 效果如图 8-16 所示。从 Nyquist 曲线可以看出, 系统没有包围 $(-1, j0)$ 点, 但是开环系统不稳定, 在右半平面上的两个极点分别为 $p=1$ 和 $p=2$, 因此闭环系统是不稳定的。从图 8-16 中的图 (b) 也可以看出, 闭环系统阶跃响应有分散趋势, 所以闭环系统是不稳定的。

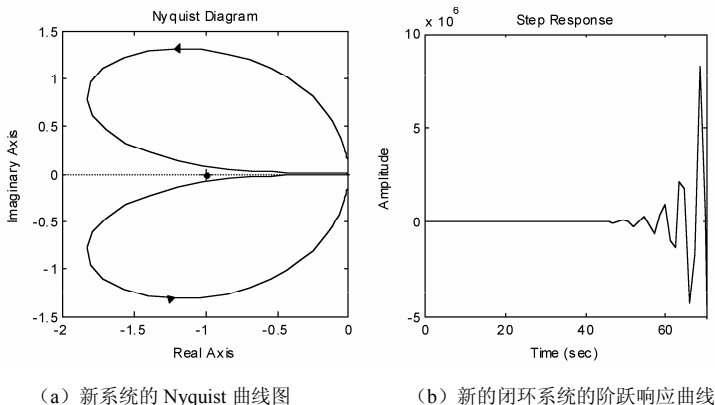


图 8-16 新系统的 Nyquist 曲线和闭环系统的阶跃响应曲线

从例 8-18 中可以看出, 通过绘制系统开环 Nyquist 曲线, 可以根据 Nyquist 稳定性判据来判断闭环系统的稳定性, 同时可以通过增加或者减少零点和极点的数目, 来改善系统的稳定性。

8.4.2 对数频率特性

1. Bode图

Bode 图是指将系统的频率特性绘制于对数坐标中。在上一节中给出了控制系统的频率特性表示形式 $G(j\omega)$, 那么将它们绘制于对数坐标时, 则取对数可得

$$\lg[G(j\omega)] = \lg[A(\omega)e^{j\varphi(\omega)}] = \lg[A(\omega)] + j\varphi(\omega)\lg e$$

令 $L(\omega) = 20\lg[A(\omega)]$, 则可以将幅频特性描述为对数频率特性 $L(\omega)$ 。使用对数频率特性研究频率范围很宽的频率特性时, 由于采用了对数坐标, 因此可以描述较大的频率带宽, 包括低频段、中频段和高频段, 这样非常便于对系统在不同频段下进行系统性能的分析。

2. 开环对数频率特性用于闭环系统稳定性判据

相对于开环幅频特性, 开环对数频率特性用于闭环系统稳定性判据有以下两条。

① 如果开环系统稳定, 即开环系统所有的极点都位于复平面的左半平面, 那么在对数幅频特性大于 0dB 的所有频段内, 对数相频特性与 $-\pi$ 线正穿越和负穿越次数之差为 0, 即闭环系统是稳定的, 否则, 闭环系统不稳定。

② 如果开环系统不稳定, 那么开环系统有极点位于复平面的右半平面, 假设有 P 个极点

位于右半平面, 那么在对数幅频特性大于 0dB 的所有频段内, 对数相频特性与 $-\pi$ 线正穿越和负穿越次数之差为 $p/2$ 时, 则闭环系统稳定, 否则, 闭环系统是不稳定的。

3. MATLAB应用示例

【例 8-19】绘制二阶系统 Bode 图并进行分析。

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

分别绘制 $\xi = 0.707$, $\omega_n = [4 \ 6 \ 8 \ 10]$ 时的 Bode 图和固有频率 $\omega_n = 5$, $\xi = [0.1 \ 0.3 \ 0.5 \ 0.707 \ 0.9]$ 时系统的 Bode 图。通过增加零点和极点, 对系统稳定性进行分析。

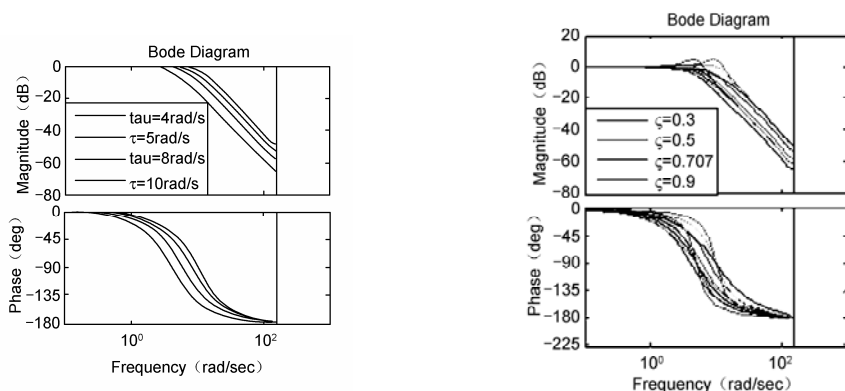
代码如下:

```
zeta=0.707;
wn=[4 6 8 10];           %系统阻尼系数, 不同角频率下系统的 Bode 图
for i=1:length(wn)
    s=tf(wn(i)^2,[1 2*zeta*wn(i) wn(i)^2]); %系统传递函数
    s=c2d(s,0.02,'foh'); %系统采样周期为 0.02s, 采样算法为 foh 进行离散化
    bode(s);               %绘制系统的 Bode 图
    dbode(s.num{1},s.den{1},0.02); %离散系统的 Bode 图
    hold on;
end
legend('tau=4rad/s','tau=5rad/s','tau=8rad/s','tau=10rad/s');
```

运行程序, 效果如图 8-17 (a) 所示。

```
wn=5;
zeta=[0.3 0.5 0.707 0.9]; %系统角频率, 不同阻尼系数下系统 Bode 图
for i=1:length(zeta)
    s=tf(wn^2,[1 2*zeta(i)*wn wn^2]); %系统传递函数
    s=c2d(s,0.02,'foh'); %系统采样周期为 0.02s, 采样算法为 foh 进行离散化
    bode(s); %绘制系统的 Bode 图
    dbode(s.num{1},s.den{1},0.02); %离散系统的 Bode 图
    hold on;
end
legend('\zeta=0.3','\zeta=0.5','\zeta=0.707','\zeta=0.9');
```

运行程序, 效果如图 8-17 (b) 所示。



(a) $\xi = 0.707$ 、 $\omega_n = [4 \ 6 \ 8 \ 10]$ 时的 Bode 图

(b) $\omega_n = 5$ 、 $\xi = [0.1 \ 0.3 \ 0.5 \ 0.707 \ 0.9]$ 时的 Bode 图

图 8-17 连续系统不同的固有频率和阻尼系数对系统的影响

取 $\xi = 0.707$, $\omega_n = 0.5$ 时, 运行下面的程序段:

```
zeta=0.707;wn=0.5;
s=tf(wn^2,[1,zeta*wn wn^2]);
f1=5+10*i;
r1=evalfr(s,f1) %计算系统在单点(复数)处的频率响应
w=[10 20 50+i*40];
r2=freqresp(s,w) %计算系统在多点向量处的频率响应
[gm,pm,wcg,wcp]=margin(s) %计算系统的增益、相角裕度和截止频率
```

运行程序, 结果如下:

```
r1 =
    1.2248e-004
r2(:,1) =
   -0.0025 - 0.0001i
r2(:,2) =
   -6.2520e-004 -1.1057e-005i
r2(:,3) =
   -5.6690e-006 -9.5428e-009i
ans=
    Inf    59.9901    Inf    0.6124
```

取 $\xi = 0.707$, $\omega_n = 0.5$ 时, 对原系统增加一个极点 $p=1$, 那么重新绘制系统的 Bode 图和 Nyquist 曲线。代码如下:

```
zeta=0.707;wn=0.5;
s=tf(wn^2,[1,zeta*wn wn^2]);
den=conv([1 -2],s.den{1});
s_new=tf(s.num{1},den);
subplot(121);bode(s);grid on;
subplot(122);bode(s_new);grid on
figure(2)
subplot(121);nyquist(s);grid on
subplot(122);nyquist(s_new);grid on
```

运行程序, 效果如图 8-18 所示, 可以看出加入一个极点 $p=1$ 后, 根据 Nyquist 稳定性判据和开环对数频率特性用于闭环系统稳定性判据均可以分析, 得出增加极点后的系统是不稳定的。有兴趣的读者可以自行增加或减少零点或极点来试试, 看看输出的效果。

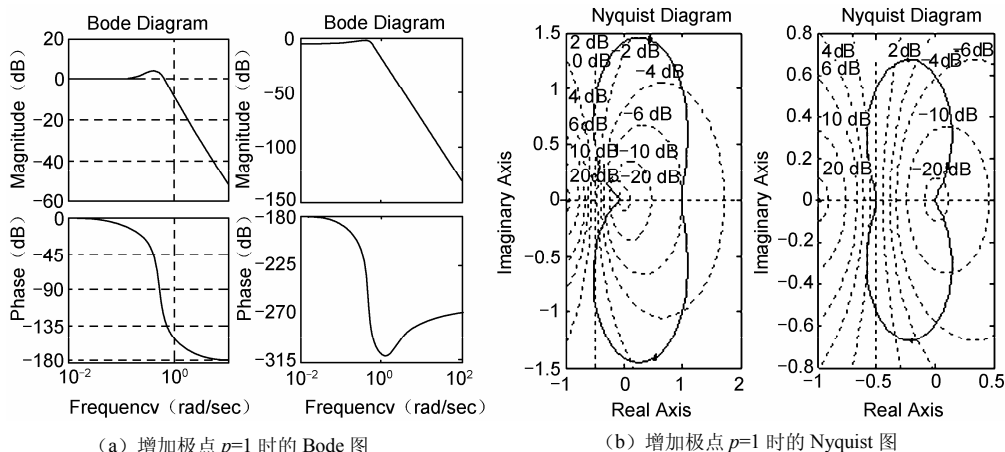


图 8-18 增加了一个极点后系统的 Bode 图和 Nyquist 图

8.4.3 对数幅相特性

对数幅相特性是指将对数幅频特性和对数相频特性绘制在一个坐标平面上，以对数幅值作为纵坐标，以相位移作为横坐标，以频率作为参变量。通常也称为尼柯尔斯图（Nichols）。在 MATLAB 控制系统工具箱中，可以使用 `nichols` 函数来绘制 Nichols 图。

【例 8-20】绘制开环系统的对数幅相频率特性。

$$G(s) = \frac{5}{s(s+5)(s+13)}$$

代码如下：

```
s=zpk([], [0 -5 -13], 5); %生成系统零极点模型
nichols(s) %绘制对数幅相特性曲线
ngrid
```

运行程序，效果如图 8-19 所示。

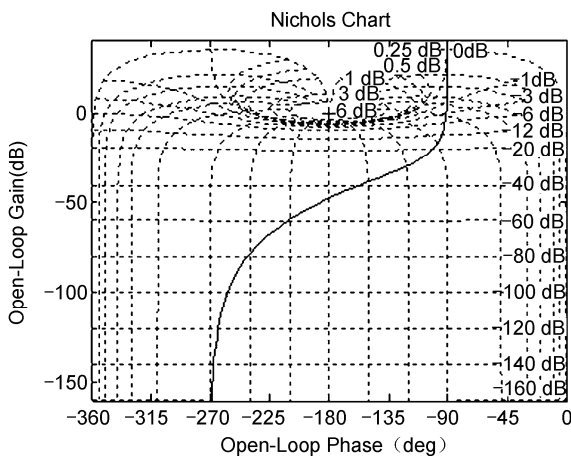


图 8-19 系统的 Nichols 图

8.5 系统校正

系统校正的方法非常多，通常可以分为串联校正、反馈校正和前馈校正。

8.5.1 串联超前校正

通过在开环控制系统中串联超前校正装置 $G_c(s)$ ，其传递函数为

$$G_c(s) = \frac{1 + \alpha Ts}{1 + Ts} \quad (\alpha > 1)$$

通过相角超前特性提高系统的相角裕度和截止频率，从而减小系统的超调量，提高系统的快速性，改善动态性能，但是由于截止频率的提高，系统抗高频干扰能力较弱。利用频率响应法进行串联超前校正的步骤如下。

- ① 根据稳态误差 e_{ss} 的要求确定系统开环增益系数 K 。
- ② 绘制校正系统对数频率特性曲线（Bode 图），计算相角裕度 γ' 和截止频率 ω'_c 。

③ 根据相角裕度的要求计算超前校正装置的相位超前量 $\phi_m = \gamma'' - \gamma' + \sigma$, 其中 σ 为用于补偿因超前校正装置引入, 使系统截止频率增大而增加的相角滞后量, 通常取值为 $5^\circ \sim 10^\circ$ 。

④ 根据所确定的最大相位超前角 ϕ_m , 计算系数 α 。

$$\alpha = \frac{1 + \sin(\phi_m)}{1 - \sin(\phi_m)}$$

⑤ 计算校正后系统的开环截止频率 ω_c'' , 由校正前系统的对数幅频特性曲线, 求得其幅值为 $-10 \lg a$ 处的频率, 即校正后系统的开环截止频率 ω_c'' 。

⑥ 根据校正后系统的开环截止频率 ω_c'' 和系数 α , 计算周期 T 。

$$T = \frac{1}{\omega_c'' \sqrt{\alpha}}$$

⑦ 画出串联超前校正网络的系数的对数频率特性, 验证相角裕度是否满足要求, 如果不满足, 重新返回③处, 增大 σ , 重新进行计算。

【例 8-21】控制系统的开环传递函数。

$$G(s) = \frac{K}{s(0.5s+1)(0.2s+1)}$$

设计串联超前校正系统, 要求系统对单位斜坡输入信号的稳态误差 $e_{ss} \leq 1\%$, 相角裕度 $\gamma \geq 25^\circ$ 。

(1) 绘制校正前系统的 Bode 图和根轨迹, 计算相角裕度。

```
clear;
s1=zpk([], [0 -2 -5], 99); %生成控制系统的零极点模型
[wnm, pm, wcg, wcp] = margin(s1); %计算截止频率和相角裕量
[wnm, pm, wcg, wcp]
subplot(121); bode(s1); %绘制系统的 Bode 图
subplot(122); rlocus(s1); %绘制根轨迹
```

计算如下:

```
ans =
    0.7071    -8.6456    3.1623    3.7389
```

校正前系统的 Bode 图和根轨迹如图 8-20 所示。

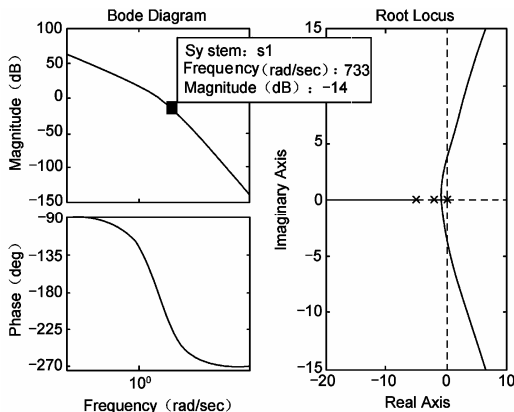


图 8-20 校正前系统的 Bode 图和根轨迹

(2) 根据系统稳态误差的要求, 确定开环增益系数。

$$K_v = \lim_{s \rightarrow 0} G_s(s)G(s) = \lim_{s \rightarrow 0} \frac{K}{1+s(0.5s+1)(0.2s+1)} = K$$

所以系统的稳态误差 e_{ss} 可以计算为

$$e_{ss} = \frac{1}{K_v} = \frac{1}{10K} \leq 1\% \Rightarrow K = 100$$

那么系统的开环传递函数可以表示为

$$G(s) = \frac{100}{s(0.5s+1)(0.2s+1)}$$

(3) 设计超前校正网络。

通过下面的代码进行超前校正网络设计:

```
m=1;step=0.5;
while (pm<=25)
    p_m=25-pm+5+m*step; %计算超前校正装置最大的超前相角
    a=(1+sin(p_m))/(1-sin(p_m)); %计算超前校正装置系统 a
    w=logspace(-2,3,1000); %生成 1000 个点的对数向量
    m_m=-10*log10(a);
    [mag,ph]=bode(s1,w); %获取传递函数 s 在给定频率 w 下对应的幅值和相位
    mag_1(1:length(mag),1)=20*log10(mag(:,1:length(mag)));
    index=find(mag_1<=m_m);
    wc_p=w(index(1)); %计算校正后系统的截止频率
    T=1/(wc_p*sqrt(a)); %计算超前校正网络的周期 T
    sc=tf([a*T 1],[T 1]); %得到超前校正网络的传递函数
    s=series(sc,s1); %校正后系统的传递函数
    [wm,pm,wcg,wcp]=margin(s) %计算校正后系统的相角裕度和截止频率
    m=m+1;
    clear mag_1;
end
[wm,pm,wcg,wcp]
figure;
subplot(121);bode(s);
subplot(122);rlocus(s);
figure(2)
bode(s1);hold on;bode(s)
```

校正后系统的相角裕度和截止频率分别是:

```
ans =
    8.4082    26.9092    33.8984    11.2535
```

校正后系统的 Bode 图的根轨迹如图 8-21 所示。

超前校正前后系统的 Bode 图对比如图 8-22 所示。

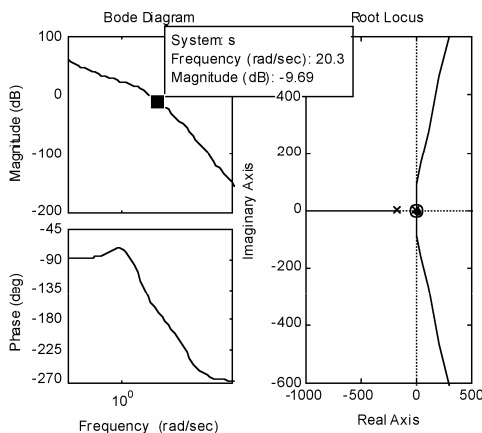


图 8-21 校正后系统的 Bode 图和根轨迹

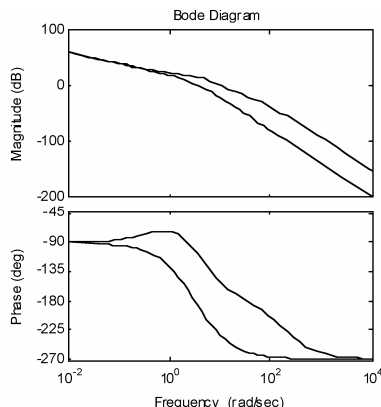


图 8-22 超前校正前后系统的 Bode 图对比

8.5.2 串联滞后校正

串联滞后校正是在开环控制系统中串联滞后校正装置 $G_c(s)$ ，其传递函数为

$$G_c(s) = \frac{1 + \beta Ts}{1 + Ts} \quad (\beta < 1)$$

串联滞后校正装置利用高频衰减特性来减小截止频率 ω_c ，提高相角裕度 γ ，从而减小系统的超调量，由于截止频率减小，那么可以有效地抑制高频噪声，但是不利于系统的快速性。

利用频率响应法进行串联滞后校正的一般步骤如下。

- ① 根据要求的稳态误差 e_{ss} 计算开环增益 K 。
- ② 绘制校正前系统开环传递函数的 Bode 图，并确定校正前系统的截止频率 ω'_c 、相角裕度 γ' 和幅值裕度 $h(\text{dB})$ 等。
- ③ 选择不同的截止频率 ω''_c ，计算对应的相角裕度 $\gamma'(\omega''_c)$ 。
- ④ 根据相角裕度 γ' 的要求，选择校正系统的截止频率 ω''_c 计算系统校正后期望的相角裕度 γ'' 。

$$\gamma'' = \gamma'(\omega''_c) + \varphi_c(\omega''_c)$$

其中， $\varphi_c(\omega''_c)$ 是考虑到滞后网络在新的截止频率 ω''_c 处所产生的相角滞后，通常取

$$\varphi_c(\omega''_c) = -6 \sim -14$$

- ⑤ 根据以下两式计算串联滞后校正装置的系数。

$$20 \lg \beta + L(\omega''_c) = 0$$

$$\frac{1}{\beta T} = (0.1 \sim 0.25) \omega''_c$$

- ⑥ 绘制校正后系统的对数幅频特性，验证相角裕度和幅值裕度是否符合要求。

接着例 8-21，设计串联滞后校正网络，使校正系统相角裕度大于 45° ，幅值裕度 $20 \lg(h) \geq 10 \text{dB}$ 。

代码如下：

```

m=1;step=-0.5;pm_r=45;
while (pm<=pm_r)
    p_c=-6+(m-1)*step; %滞后网络的截止频率处所产生的相角滞后
    w=logspace(-3,4,1000); %生成 1000 个点的对数向量
    ga=90-180*(atan(0.2*w)+atan(0.5*w))/pi;
    ga_p=pm_r-p_c;
    index=find(ga<=ga_p);
    wc_p=w(index(1)); %计算截止频率
    [mag,pha]=bode(s1,wc_p);
    b=10^(-log10(mag)); %计算系数 b
    T=1/(wc_p*0.1*b); %计算系数 T
    sc=tf([b*T 1],[T 1]);
    s=series(sc,s1);
    [wm,pm,wcg,wcp]=margin(s);
    m=m+1;
    clear mag_1;
end
[wm,pm,wcg,wcp]
figure;
subplot(121);bode(s);
subplot(122);rlocus(s);
figure(2)
bode(s1);hold on;bode(s)

```

校正后系统的相角裕度和截止频率分别是：

```

ans =
    5.4218    45.4073    3.0586    1.0512

```

校正后系统的 Bode 图和根轨迹如图 8-23 所示。

校正前后系统的 Bode 图对比如图 8-24 所示。

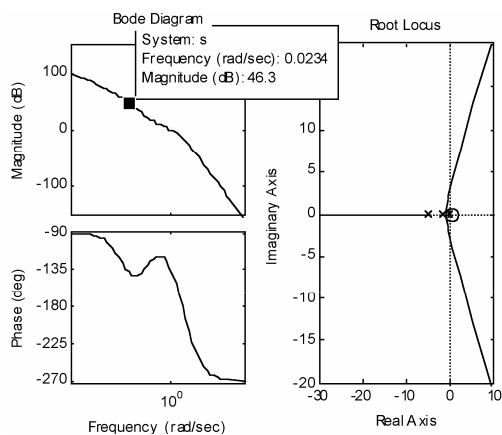


图 8-23 校正后系统的 Bode 图和根轨迹

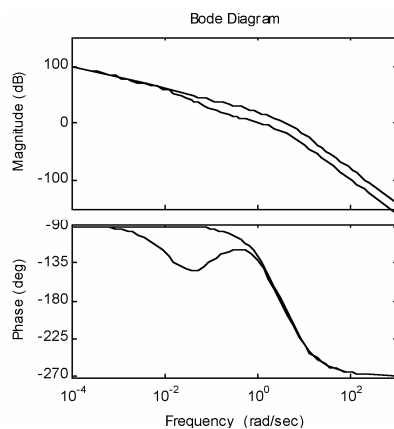


图 8-24 校正前后系统的 Bode 图对比

8.5.3 串联滞后—超前校正

滞后—超前校正网络兼有滞后校正和超前校正的特点，它利用滞后校正网络的幅值衰减特点提供系统的稳态精度，抑制高频噪声，同时用超前校正网络提高系统的相角裕度和截止频率，从而提高系统的快速性，改善系统的动态性能指标。进行串联滞后—超前校正设计的一般步骤如下。

- ① 根据要求的稳态误差 e_{ss} 计算开环增益 K 。
- ② 绘制校正前系统开环传递函数的 Bode 图, 并确定校正前系统的截止频率 ω'_c 、相角裕度 γ' 和幅值裕度 $h(\text{dB})$ 等。
- ③ 在校正前系统的对数幅频曲线上, 选择 -20dB 线和 -40dB 线的交点频率作为网络超前部分的交点频率 ω_p 。
- ④ 由要求的截止频率 ω_c'' , 可以确定系数 α 。

$$-10\lg \alpha + L(\omega_c'') + 20\lg(T_p \omega_c'') = 0$$

- ⑤ 由相角裕度要求计算网络滞后部分的交点频率 ω_α 。
- ⑥ 绘制校正后系统的对数频率特性, 验证性能指标。

【例 8-22】单位反馈系统开环传递函数 $G(s) = \frac{K}{s(0.1s+1)(0.02s+1)}$, 设计串联校正装置, 要求系统对单位斜坡输入信号的稳态误差 $e_{ss} \leq 1\%$, 相角裕度 $\gamma' \geq 40^\circ$, 截止频率 $\omega_c'' = 18\text{rad/s}$ 。

(1) 绘制校正前系统的 Bode 图和根轨迹, 计算相角裕度和截止频率。

和例 8-21 相似, 首先绘制开环系统的 Bode 图和根轨迹, 计算校正前系统的截止频率和相角裕度, 选择合适的校正装置形式。

```
clear;
s1=zpk([], [0 -10 -50], 100*10*50); %建立零极点模型
[wm,pm,wcg,wcp]=margin(s1); %计算校正前系统的截止频率和相角裕度
[wm,pm,wcg,wcp]
subplot(121);bode(s1);
subplot(122);rlocus(a);
```

运行结果为:

```
ans =
    0.6000   -10.5320   22.3607   28.6233
```

可以看出, 校正前系统的截止频率 $\omega'_c = 28.6233\text{rad/s}$, 相角裕度 $\gamma' = -10.5320$, 所以该系统不稳定, 而且截止频率比要求的截止频率 ω_c'' 要大, 因此采用串联滞后—超前校正网络。校正前系统的 Bode 图和根轨迹如图 8-25 所示。

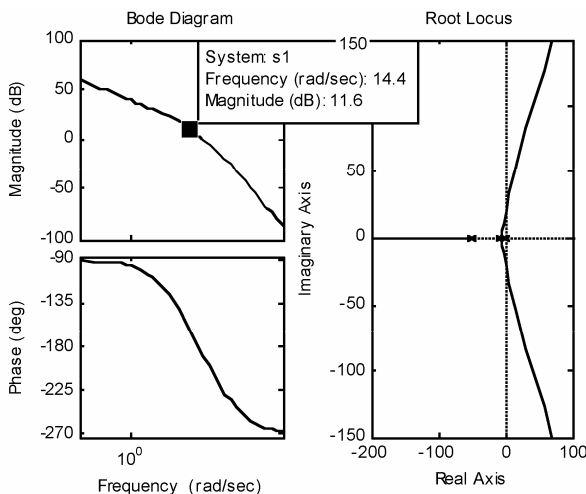


图 8-25 校正前系统的 Bode 图和根轨迹

(2) 根据系统稳态误差要求, 确定开环增益系数。

根据系统稳态误差的要求, 可以计算得到 $K=100$ 。

(3) 串联滞后—超前校正网络设计。

首先在图 8-23 中计算校正前系统 -20dB 和 -40dB 的交接频率 $\omega_b=10\text{rad/s}$ 。所以 $T_b = \frac{1}{\omega_b} = 0.1$ 。

根据校正后系统要求 $\omega_c'' 18\text{rad/s}$, 那么可以得到系数 $\alpha = 6.4441$ 。

```
wc_p=18;
wb=10;
[mag,ph]=bode(s1,wc_p); %获取 Bode 图上 wc=18rad/s 处的幅值和相位
T=1/wb;
alpha=10^(-20*log10(mag)/(10));
```

最后确定校正系统滞后部分的交接频率 ω_a 。给定的滞后—超前校正网络

$G_c(s) = \frac{(s + \omega_a)(s + \omega_b)}{(s + \omega_a/\alpha)(s + \alpha\omega_a)}$, 其中 $\omega_b = 10$, $\alpha = 6.4441$, 在校正后的网络传递函数中代

入截止频率 $\omega_c'' = 18\text{rad/s}$ 时, $\gamma'' = 40^\circ$ 可以计算得出 ω_a :

```
>> solve('90+180*(atan(18/x)-atan(0.02*18)-atan(6.4441*18/x)-atan(18/6.4441))/pi=40')
```

运行可以得到:

```
ans =
-94.202485968205246790246794250531
-22.163835471440675981693629495398
```

取 $\omega_a = 5.6301\text{rad/s}$ 。于是就得到校正滞后—超前网络的传递函数, 并绘制校正后系统的 Bode 图和根轨迹。代码如下:

```
wa=5.6301;
sc=zpk([-wa -wb],[-wa/alpha -alpha*wb],1);
s=series(sc,s1);
[wm,pm,wcg,wcp]=margin(s);
[wm,pm,wcg,wcp]
subplot(121);bode(s);
subplot(122);rlocus(s);
figure(2)
bode(s1);hold on;bode(s)
```

输出结果为:

```
ans =
2.5159 24.4617 38.2656 22.1419
```

滞后—超前校正后系统的 Bode 图和根轨迹如图 8-26 所示。

滞后—超前校正前后系统的 Bode 图对比如图 8-27 所示。

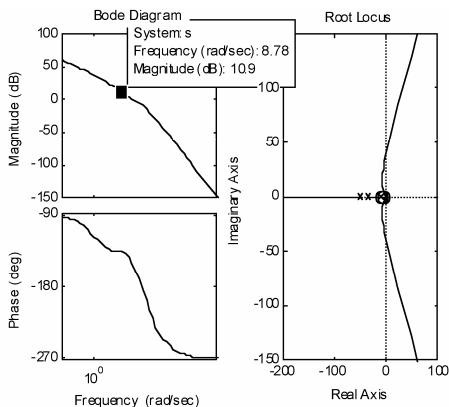


图 8-26 滞后—超前校正后系统的 Bode 图和根轨迹

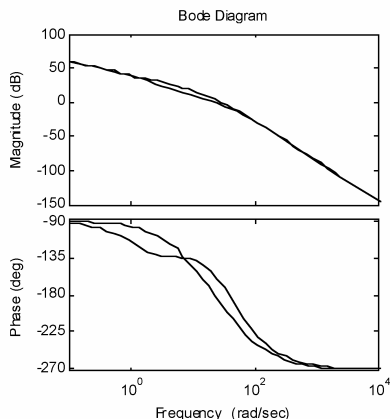


图 8-27 滞后—超前校正前后系统的 Bode 图对比

8.6 极点配置设计方法

极点配置方法是控制系统时域设计理论中最早发展起来的一种设置思想，经过在工程实际中长期的发展与完善，现在已经有一套完备的理论和丰富的工程设计实际经验。

给定控制系统的状态方程模型，希望引入某种控制，使得该系统闭环极点移动到指定位置，因为在很多情况下系统的极点位置决定系统的动态性能。以下只讨论单变量系统的极点配置问题。

设系统的模型如下所示

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

式中， A 、 B 、 C 均已给定，当引入状态反馈后，则系统的控制信号为 $u = r + K^T x$ ，这里 r 为系统的外部参考输入， K 为一列矢量，此时闭环系统的状态方程模型可以写为如下形式

$$\begin{cases} \dot{x} = (A + BK^T)x + Br \\ y = Cx \end{cases}$$

可以证明若给定系统是可控的，则可以通过状态反馈将该系统的闭环极点进行任意配置。本书略去证明过程，有兴趣的读者请参考相关资料。

8.6.1 Gura-Bass算法

假定期望的闭环系统极点为 μ_i ， $i = 1, \dots, n$ ，则原系统的开环特征方程 $\alpha(s)$ 和闭环系统的特征方程 $\beta(s)$ 分别可以写成如下形式

$$\alpha(s) = \det(sI - A) = s^n + \sum_{i=0}^{n-1} \alpha_i s^i$$

$$\beta(s) = \prod_{i=1}^n (s - \mu_i) = s^n + \sum_{i=0}^{n-1} \beta_i s^i$$

若原系统可控，则状态反馈矢量 \mathbf{K} 可由下式求出

$$\mathbf{K}^T = (\alpha - \beta)^T \mathbf{L} \mathbf{C}^{-1}$$

式中各参量应满足

$$\begin{aligned} (\alpha - \beta)^T &= [(\alpha_0 - \beta_0), \dots, (\alpha_{n-1} - \beta_{n-1})] \\ \mathbf{C} &= [\mathbf{B}, \mathbf{A}\mathbf{B}, \dots, \mathbf{A}^{n-1}\mathbf{B}] \\ \mathbf{L} &= \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} & 1 \\ \alpha_2 & \alpha_3 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ \alpha_{n-1} & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \quad (\mathbf{L} \text{ 是一个汉尔克矩阵}) \end{aligned}$$

可见若原系统可控，则 \mathbf{C}^{-1} 存在。另外可证明 \mathbf{L} 非奇异，故可以任意地配置原系统。

8.6.2 Ackermann配置算法

问题的描述与 Gura-Bass 算法相同，但 Ackermann 配置算法如下所示

$$\mathbf{K}^T = -[0 \ 0 \ \cdots \ 0 \ 1] \mathbf{C}^{-1} \beta(\mathbf{A})$$

其中 \mathbf{K} 为待求的状态反馈矢量。

【例 8-23】系统的开环状态方程为

$$\dot{\mathbf{x}} = \begin{bmatrix} -2 & -1 & 1 \\ 1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u$$

容易验证，本系统是完全可控的，并且原系统有不稳定极点。为验证上面的结论，代码如下：

```
A=[-2 -1 1;1 0 1;-1 0 1];
B=[1 1 1]';
r=rank([B A*B A*A*B])
e=eig(A)
```

运行后得到结果为：

```
r =      3
e =
-1.0000 + 1.0000i
-1.0000 - 1.0000i
1.0000
```

由上面的结果可见， $\text{rank}[\mathbf{B} \ \mathbf{A}\mathbf{B} \ \cdots \ \mathbf{A}^{n-1}\mathbf{B}] = n$ ，故系统可控；另外系统有一个特征根为

1, 所以系统不稳定。下面进行极点配置, 用 Gura-Bass 算法将闭环极点配置到-1, -2, -3, 代码如下:

```
ch_A=poly(A); %计算  $\alpha(s)$ 
ch_N=conv([1 1],conv([1 2],[1 3])); %计算  $\beta(s)$ 
i1=length(ch_A):-1:2;
diff_A=ch_A(i1)-ch_N(i1); %计算  $(\alpha-\beta)$ 
c1=B;
b1=B;
for i=2:length(A)
    b1=A*b1;
    c1=[c1 b1];
end
L=hankel([ch_A(length(ch_A)-1:-1:2)';1]); %计算汉尔克矩阵
K1=diff_A*inv(L)*inv(c1)
```

运行结果:

```
K1 =
    1.0000    -2.0000   -4.0000
```

即状态反馈矢量为

$$\mathbf{K} = \begin{bmatrix} 1 \\ -2 \\ -4 \end{bmatrix}$$

下面再验证一下配置结果, 执行下面代码:

```
>> eig(A+B*K1)'
ans =
   -1.0000   -2.0000   -3.0000
```

可见已经将极点配置到-1, -2, -3。

下面用 Ackermann 配置算法同样将极点配置到-1, -2, -3, 执行下面的代码:

```
>> eig(A+B*K1)'
K2=[zeros(1,length(A)-1),1]*inv(c1)*polyvalm(ch_N,A)
```

运行后得到结果:

```
K2 =
    -1     2     4
```

可见两种方法得到的结果是相同的。

MATLAB 控制工具箱中提供了极点配置函数 place()和 acker(), 其调用格式是相同的, 如下所示:

```
K=place(A, B, p)
K=acker(A, B, p)
```

其中, p 是指定极点的位置列矢量。

调用这两个函数可以直接得到与前面相同的结果，如下所示：

```
p=[-1 -2 -3]';  
R1=place(A,B,p)  
R2=acker(A,B,p)
```

运行后结果：

```
R1 =  
-1.0000    2.0000    4.0000  
R2 =  
-1     2     4
```

第 9 章 MATLAB在数字信号中的应用

随着 MATLAB 的出现和不断完善，尤其是 MATLAB 信号分析工具箱的推出，越来越多的电信工程师们已经意识到用 MATLAB 来解决电信工程中的实际问题是一种省时又省力的选择。由于 MATLAB 所具有的计算快速、准确和使用方便等优点，在过去的 10 年中，MATLAB 已经成为数字信号处理（Digital Signal Processing, DSP）应用中分析和设计的主要仿真工具。

9.1 数字信号知识

信号的产生是数字信号处理最基础的内容。表 9-1 列出了 MATLAB 信号处理工具箱中的一系列信号产生函数。

表 9-1 信号产生函数

函 数	产生信号类型	函 数	产生信号类型
square(t, DUTY)	方波信号	rectpuls	非周期方波信号
sawtooth	锯齿波信号	tripuls	非周期三角波信号
sinc	Sinc 信号	pulstran	脉冲序列
diric	Dirichlet 信号	chirp	调频余弦信号
rand	均匀分布白噪声信号	randn	高斯分布白噪声信号

9.1.1 信号产生

1. 周期信号

下面通过示例来说明产生周期信号的各函数。其中各函数的调用格式请注意示例中的程序。

【例 9-1】产生一个周期为 1 的方波信号，其占空比为 60%。

代码如下：

```
t=0:0.01:6;
Duty=60; %占空比为 60%
x=square(2*pi*t,Duty); %产生周期为 1，占空比为 60%的方波
plot(t,x);
axis([0 6 -1.1 1.1]);
title('Square wave');
```

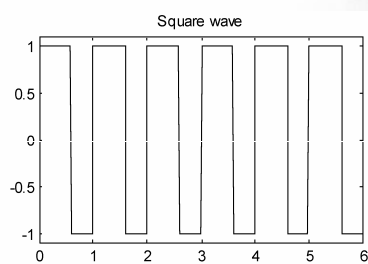


图 9-1 方波信号波形

运行程序，效果如图 9-1 所示。

【例 9-2】分别产生一个周期为 1 的锯齿波和标准对称的三角波信号。

代码如下：

```
t=0:0.01:6;
width=0.5;
x1=sawtooth(2*pi*t); %产生锯齿波
x2=sawtooth(2*pi*t,width); %产生三角波
```

```
subplot(211);plot(t,x1);
axis([0 6 -1.1 1.1]);
title('Sawtooth wave');
subplot(212);plot(t,x2);
axis([0 6 -1.1 1.1]);
title('Triangular wave');
```

运行程序，效果如图 9-2 所示。

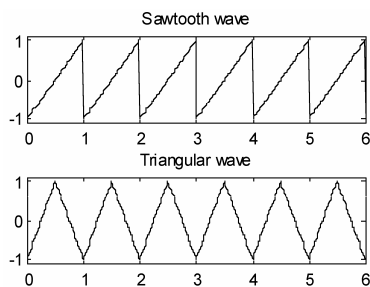


图 9-2 锯齿波和三角波信号波形

2. 脉冲信号

【例 9-3】产生一个宽度为 0.6s 的非周期方波信号。

代码如下：

```
t=-1:0.01:1;
width=0.6;
x=rectpuls(t,width); %绘制非周期方波
plot(t,x);
axis([-1 1 -0.1 1.1]);
title('Aperiodic square wave');
```

运行程序，效果如图 9-3 所示。

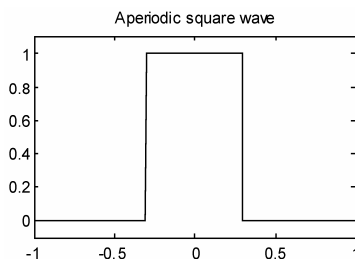


图 9-3 宽度为 0.6s 的非周期方波



注意 `rectpuls` 函数产生的方波信号非零幅值区间在左边为闭区间，在右边为开区间。比如在例 9-3 中，方波信号在 -0.3 处为 0，而在 0.3 处为 1。

【例 9-4】产生一个宽度为 1 的非周期三角波信号，并观察参数的变化对三角波形状的影响。

代码如下：

```
t=-1:0.01:1;
s1=0;s2=1;s3=-1; %s 值选取 -1,0,1
x1=tripuls(t,1,s1);
x2=tripuls(t,1,s2);
```

```

x3=tripuls(t,1,s3);
plot(t,x1,'-');hold on; %用虚线画出 s=0 时的波形
plot(t,x2,'-');hold on; %用实线画出 s=1 时的波形
plot(t,x3,'-'); %用圆点线画出 s=-1 时的波形
title('Aperiodic triangular wave');
legend('s=0','s=1','s=-1');

```

运行程序，效果如图 9-4 所示。

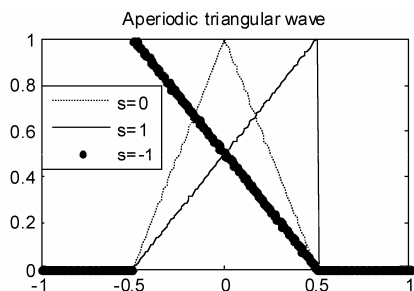


图 9-4 参数对三角波信号形状的影响

【例 9-5】按照以下要求产生指定信号：

(1) 产生一个长为 1s、采样频率为 1kHz、斜率为-1 的不对称三角波脉冲串，其中三角波幅值为 1，宽度为 0.1s，脉冲重复频率为 3Hz；

(2) 产生一个长为 0.01s、采样频率为 50kHz 的高斯脉冲串信号，其中高斯信号的中心频率为 1kHz，带宽为 50%，脉冲串的重复频率为 1kHz，重复时幅值每次衰减为上一次的 80%。

代码如下：

```

t1=0:1/1e3:1; %1kHz 采样频率，长为 1s
d1=0:1/3:1; %3Hz 脉冲重复频率
y1=pulstran(t1,d1,'tripuls',0.1,-1); %三角波幅值为 1，宽度为 0.1s
subplot(211);
plot(t1,y1);
title('三角波脉冲串');
t2=0:1/50e3:10e-3; %50kHz 采样频率，长为 0.01s
d2=[0:1/1e3:10e-3;0.8^(0:10)]'; %1kHz 脉冲重复频率，幅值每次衰减为 80%
y2=pulstran(t2,d2,@gauspuls,10e3,0.5); %高斯信号的中心频率为 1kHz，带宽为 50%
subplot(212);
plot(t2,y2);
title('高斯信号脉冲串');

```

运行程序，效果如图 9-5 所示。

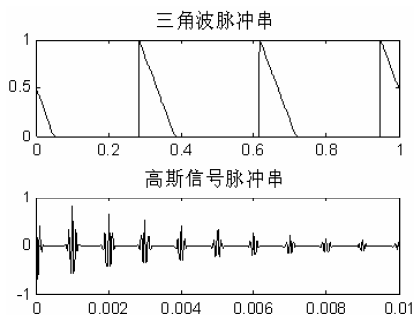


图 9-5 三角波脉冲串和高斯脉冲串

3. 特殊信号

(1) Sinc 信号

Sinc 信号是数字信号处理中的一个很重要的信号，因为它的傅里叶变换正好是幅值为 1 的矩形脉冲，它的表达式如下

$$\text{Sinc}(t) = \begin{cases} 1, & t = 0 \\ \frac{\sin(\pi t)}{\pi t}, & t \neq 0 \end{cases}$$

其调用格式请看以下示例。

【例 9-6】产生一个典型的 Sinc 信号，并画出其波形。

代码如下：

```
t=-5:0.01:5;
x=sinc(t);
plot(t,x);
title('Sinc wave');
```

运行程序，效果如图 9-6 所示。

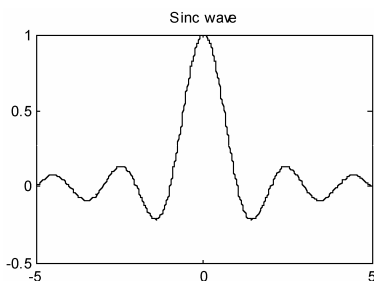


图 9-6 Sinc 信号波形

(2) Dirichlet 信号（周期 Sinc 函数）

Dirichlet 信号又称周期 Sinc 信号，其表达式为 $y=\text{diric}(x,n)$ 。

调用 MATLAB 中的 diric 函数可以产生 Dirichlet 信号，其表达式如下

$$\text{diric}(x,n) = \begin{cases} (-1)^{\frac{x}{2\pi}(n-1)}, & x = 0, \pm 2\pi, \pm 4\pi, \dots \\ \frac{\sin(nx/2)}{n\sin(x/2)}, & \text{其他} \end{cases}$$

其调用格式请参看以下示例。

【例 9-7】产生 $0 \sim 6\pi$ 之间的 Dirichlet 信号波形，并分析 N 的奇偶性对波形周期的影响。

代码如下：

```
x=0:0.01:6*pi;
y1=diric(x,11);
y2=diric(x,12);
subplot(211);
plot(x,y1);
title('Diric wave(N=11)');
subplot(212);
plot(x,y2);
```

```
title('Diric wave(N=12)');
```

运行程序，效果如图 9-7 所示。从中可以看出，当 N 为奇数 ($N=11$) 时，所产生的 Dirichlet 信号周期为 2π ；当 N 为偶数 ($N=12$) 时，所产生的 Dirichlet 信号周期为 4π 。

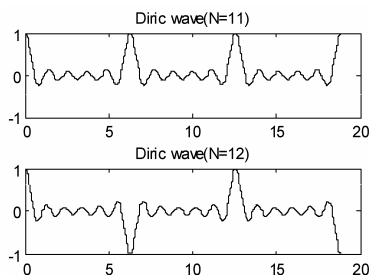


图 9-7 N 的奇偶性对 Dirichlet 信号周期的影响

4. 随机信号

在随机信号的处理中，经常要用到白噪声（一种随机噪声）的模型，MATLAB 也提供了相应的函数来产生白噪声信号。这里介绍 rand 函数和 randn 函数。

(1) rand 函数

rand 函数用来产生均值为 0.5，方差为 1/12，幅值在 0~1 之间变化的伪随机数，在信号处理中常用它来模拟均匀分布的白噪声信号。

(2) randn 函数

randn 函数用来产生均值为 0，方差为 1，服从高斯分布的伪随机数，信号处理中常用它来模拟满足高斯分布的白噪声。

这两个函数的调用格式相似，请读者参看以下示例。

【例 9-8】产生一个均值为 0，方差为 0.02 的均匀分布的白噪声，并画出其波形。

代码如下：

```
x=0:0.01:6*pi;
y1=diric(x,11);
y2=diric(x,12);
subplot(211);
plot(x,y1);
title('Diric wave(N=11)');
subplot(212);
plot(x,y2);
title('Diric wave(N=12)');
>> p=0.02;          %白噪声的方差
N=50000;            %白噪声序列的点数
u1=rand(1,N);        %产生均值为 0.5，方差为 1/12 的白噪声序列
u1_m=mean(u1);       %求均值
u1_v=var(u1);        %求方差
u=u1-u1_m;           %调整白噪声序列的均值，使之变为 0
u=u*sqrt(p/u1_v);    %调整白噪声序列的幅值，以调整其方差，使之变为 0.02
subplot(211);
plot(u(1:99));grid on;
title('均匀分布白噪声');
```



```
subplot(212);
hist(u,50);
title('白噪声的直方图');
u_m=mean(u)
v_v=var(u)
运行代码结果为:
u_m =
    5.1096e-016
v_v =
    0.0200
```

所得白噪声序列的波形（仅取 99 点）如图 9-8 所示。

【例 9-9】产生均值为 0，方差为 0.2，服从高斯分布的白噪声信号。

代码如下：

```
p=0.02;           %白噪声的方差
N=50000;          %白噪声序列的点数
u1=randn(1,N);    %产生均值为 0.5，方差为 1/12 的白噪声序列
u1_m=mean(u1);    %求均值
u1_v=var(u1);     %求方差
u=u1-u1_m;        %调整白噪声序列的均值，使之变为 0
u=u*sqrt(p/u1_v); %调整白噪声序列的幅值，以调整其方差，使之变为 0.2
subplot(211);
plot(u(1:99));grid on;
title('高斯分布白噪声');
subplot(212);
hist(u,99);
title('白噪声的直方图');
u_mn=mean(u)
v_vn=var(u)
运行代码结果为:
u_mn =
    3.1963e-018
v_vn =
    0.0200
```

所得白噪声序列的波形（仅取 99 点）如图 9-9 所示。

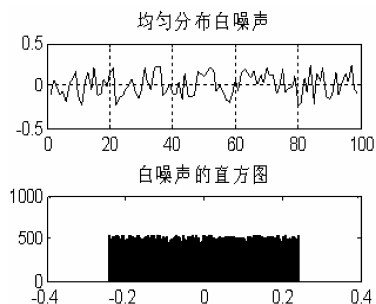


图 9-8 均值为 0，方差为 0.02 的均匀白噪声信号

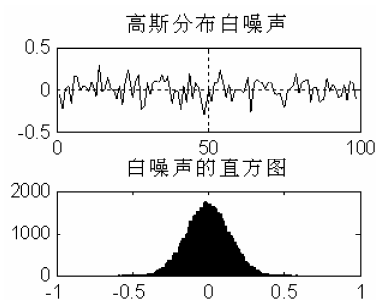


图 9-9 均值为 0，方差为 0.2 的高斯白噪声信号

9.1.2 信号的运算

在数字信号处理中,信号的运算包括加法、乘法、位乘、移位、采样和、采样积、能量、功率、卷积、相关等。下面讲解如何通过 MATLAB 来进行这些操作。由于它们的 MATLAB 实现相对比较简单,这里仅仅给出卷积和相关示例。

1. 加法

加法的数学描述为: $x(n) = x_1(n) + x_2(n)$ 。其 MATLAB 实现为: $x = x1 + x2$ 。

2. 乘法

乘法的数学描述为: $x(n) = x_1(n)x_2(n)$ 。其 MATLAB 实现为: $x = x1 * x2$ 。



加法与乘法中的 $x1$ 和 $x2$ 应当具有相同的长度,位置对应才能相加或相乘;否则,需要先通过 zeros 函数左右补零才能相加或相乘。

3. 倍乘

倍乘的数学描述为: $y(n) = a \times x(n)$ 。其 MATLAB 实现为: $y = a * x$ 。

4. 移位

移位的数学描述为: $y(n) = x(n - k)$ 。其 MATLAB 实现为: $y(n) = x(n - k)$ 。

5. 翻转

翻转的数学描述为: $y(n) = x(-n)$ 。其 MATLAB 实现为: $y(n) = \text{fliplr}(n)$ 。

6. 采样和

采样和的数学描述为: $y = \sum_{n=n_1}^{n=n_2} x(n)$ 。其 MATLAB 实现为: $y = \text{sum}(x(n1:n2))$ 。

7. 采样积

采样积的数学描述为: $y = \prod_{n=n_1}^{n=n_2} x(n)$ 。其 MATLAB 实现为: $y = \text{prod}(x(n1:n2))$ 。

8. 能量

信号能量的数学描述为: $E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2$ 。其 MATLAB 实现为: $E_x = \text{sum}(\text{abs}(x).^2)$ 。

9. 功率

信号功率的数学描述为: $P_x = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2$ 。其 MATLAB 实现为: $P_x = \text{sum}(\text{abs}(x).^2)/N$ 。

10. 卷积

信号卷积的数学描述为: $y(n) = x(n) \otimes h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$ 。其 MATLAB 实现为: $y = \text{conv}(x, h)$ 。

11. 相关

相关包括信号的互相相关和自相关,它们的数学描述如下。

自相关: $r_x(m) = \sum_{n=-\infty}^{\infty} x^*(n)x(n+m)$ 。互相关: $r_{xy}(m) = \sum_{n=-\infty}^{\infty} x^*(n)y(n+m)$ 。

其 MATLAB 实现如下。

(1) $rxy=xcorr(y,x)$: 用来求序列 x 与序列 y 的互相关。若 x 与 y 的长度均为 N , 则 rxy 的长度为 $2N-1$; 若 x 与 y 的长度不相等, 则函数先将短的那个序列补零再进行互相关。

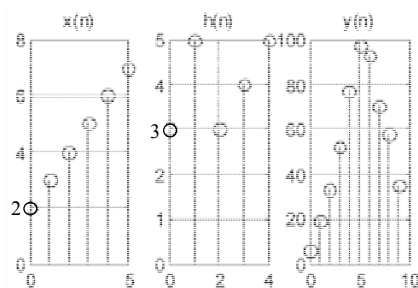
(2) $rx=xcorr(x,Mlag,'flag')$: 用来求序列 x 的自相关, $Mlag$ 表示 rx 的单边长度, 总的长度为 $2Mlag+1$; $flag$ 是定标标志, 若 $flag=biased$, 表示有偏估计, 则输出都除以序列 x 的长度为 N ; 若 $flag=unbiased$, 表示无偏估计, 则输出都除以 $(N-abs(m))$; 若 $flag$ 默认, 则 rx 不定标。

【例 9-10】令 $x(n)=[2, 3, 4, 5, 6, 7]$, $h(n)=[3, 5, 3, 4, 5]$, 求 $x(n)$ 和 $h(n)$ 的卷积 $y(n)$ 。

代码如下:

```
x=[2 3 4 5 6 7];
h=[3 5 3 4 5];
n=6;m=5;a=m+n-1;
nx=0:n-1;nh=0:m-1;ny=0:a-1;
y=conv(x,h);
subplot(131);stem(nx,x);
title('x(n)');grid on;
subplot(132);stem(nh,h);
title('h(n)');grid on;
subplot(133);stem(ny,y);
title('y(n)');grid on;
```

运行程序, 效果如图 9-10 所示。



```

title('求 y 与 x 的互相关');grid on;
subplot(223);stem(N,rx);
title('求 x 的自相关');grid on;
subplot(224);stem(M,ry);
title('求 y 的自相关');grid on;

```

运行程序，效果如图 9-11 所示。

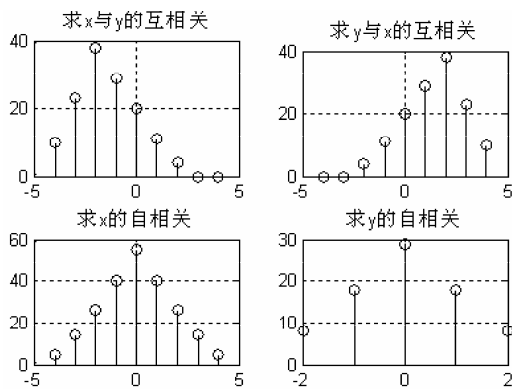


图 9-11 互相关与自相关函数

9.1.3 信号的抽取与插值

1. 信号的抽取

假设信号 $x(n)$ 的抽样率为 f_s ，现在希望对其进行 M 倍的抽取，使其抽样频率降低为 $\frac{f_s}{M}$ 。一个最简单的办法是在 $x(n)$ 中每隔 M 个点抽取一个点，以此组成一个新的序列 $x'(n)$

$$x'(n)=x(Mn), n=-\infty \sim \infty$$

为了防止抽取后 $x'(n)$ 的频谱发生混叠，一般需要先对 $x(n)$ 进行低通滤波，压缩其频带，再进行抽取。理想低通滤波器 $h(n)$ 的频率响应为

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| \leq \frac{\pi}{M} \\ 0, & \text{其他} \end{cases}$$

MATLAB 中提供了 decimate 函数进行信号的抽取，其调用格式请看以下示例。

【例 9-12】对信号 $x(n) = \cos(2\pi f n / f_s)$ ， $f / f_s = 1/40$ 进行 4 倍的抽取。

代码如下：

```

n=0:120;
x=cos(2*pi*1/40*n);
y=decimate(x,4);
subplot(211);stem(x);
axis([0 120 -1 1]);title('原始信号');grid on;
subplot(212);stem(y);
axis([0 30 -1 1]);title('4 倍抽取后的信号');grid on;

```

运行程序，效果如图 9-12 所示。

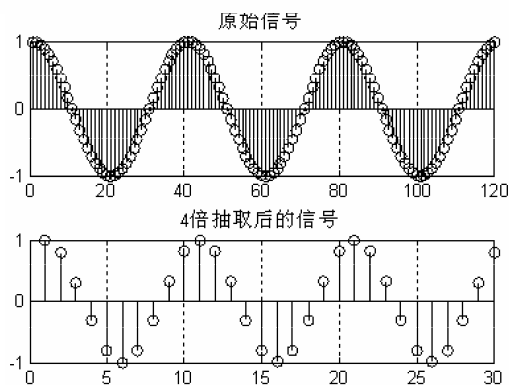


图 9-12 信号的抽取

2. 信号的插值

与信号的抽取对应，假定信号 $x(n)$ 的抽取率为 f_s ，现在通过 L 倍的插值，将其抽样率变为 Lf_s 。一个最简单的方法就是在 $x(n)$ 的每相邻两个点之间补上 $L-1$ 个零

$$v(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \dots \\ 0, & \text{其他} \end{cases}$$

然后再对信号 $v(n)$ 进行低通滤波处理，目的是为了去除 $v(n)$ 频谱的镜像。理想低通滤波器 $h(n)$ 的频率响应为

$$H(e^{j\omega}) = \begin{cases} L, & |\omega| \leq \frac{\pi}{L} \\ 0, & \text{其他} \end{cases}$$

MATLAB 提供了 `interp` 函数来进行信号的插值处理，其调用格式请参看以下示例。

【例 9-13】对信号 $x(n) = \cos(2\pi f n / f_s)$ ， $f / f_s = 1/10$ 进行 4 倍的插值。

代码如下：

```
n=0:30;
x=cos(2*pi*1/10*n);
y=interp(x,5);
subplot(211);stem(x);
axis([0 30 -1 1]);title('原始信号');grid on;
subplot(212);stem(y);
axis([0 120 -1 1]);title('4 倍插值后的信号');grid on;
```

运行程序，效果如图 9-13 所示。

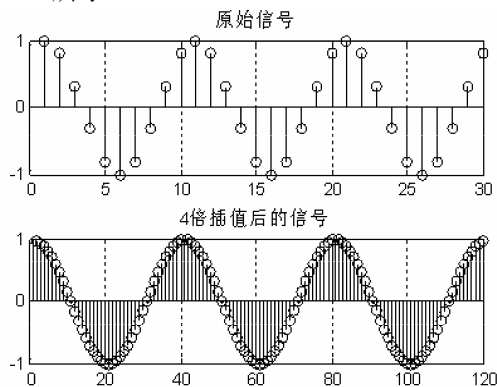


图 9-13 信号的插值

3. 信号抽取率的转换

给定一个信号 $x(n)$ ，其抽样率为 f_s ，现在需要将其抽样率变为 $f_s \frac{L}{M}$ ，这就实现了任意倍数抽样率的转换。可以通过先对信号 $x(n)$ 进行 L 倍的插值，再进行 M 倍的抽取来实现。

MATLAB 中提供了 `resample` 函数来进行抽样率的转换，其调用格式请参看以下示例。

【例 9-14】信号 $x(n) = \cos(2\pi f n / f_s)$ ， $f / f_s = 1/20$ ，将其抽样率变为原来的 3 倍。

代码如下：

```
n=0:60;
x=cos(2*pi*1/10*n);
y=resample(x,6,2);
subplot(211);stem(x);
axis([0 60 -1 1]);title('原始信号');grid on;
subplot(212);stem(y);
axis([0 150 -1 1]);title('3 倍抽样率转换后的信号');grid on;
```

运行程序，效果如图 9-14 所示。

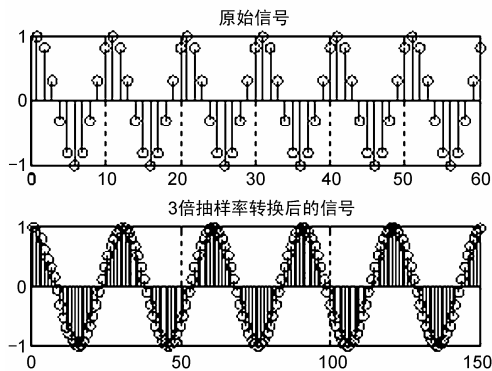


图 9-14 信号的任意倍数抽样率转换

9.2 离散时间傅里叶变换

9.2.1 离散时间傅里叶变换定义及计算

如果 $x(n)$ 是绝对可加的，即 $\sum_{n=-\infty}^{\infty} |x(n)| < \infty$ ，则其离散傅里叶变（DTFT）换定义为

$$X(e^{j\omega}) = F[x(n)] = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \quad (9-1)$$

$X(e^{j\omega})$ 的离散时间傅里叶逆变换（IDTFT）可以表示为

$$x(n) = F^{-1}[X(e^{j\omega})] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega \quad (9-2)$$

离散时间傅里叶变换有两个重要的特性。

- ① 周期性：离散时间傅里叶变换 $X(e^{j\omega})$ 是 ω 的周期函数，其周期为 2π 。
- ② 对称性：对于实值的 $x(n)$ ， $X(e^{j\omega})$ 是共轭对称的。

【例 9-15】已知序列 $x(n) = \{1, 2, \sqrt{3}, 4, 5, 6\}$ ，求 $x(n)$ 的离散时间傅里叶变换，在 $[0, \pi]$ 之间取 499 个等间隔频点，进行数值计算。

代码如下：

```
n=-2:3; %输入序列 x(n)
x=1:6;
k=0:499; %把[0,pi]分为 499 个点
w=(pi/500)*k;
r=k/500;
%进行 DTFT
X=x*(exp(-j*pi/500)).^(n*k);
magX=abs(X);
angX=angle(X);
realX=real(X);
imagX=imag(X);
%绘图
subplot(221);plot(r,magX);
grid on;title('幅度部分');
subplot(222);plot(r,angX);
grid on;title('相位部分');
subplot(223);plot(r,realX);
grid on;title('实部');
subplot(224);plot(r,imagX);
grid on;title('虚部');
```

运行程序，效果如图 9-15 所示。

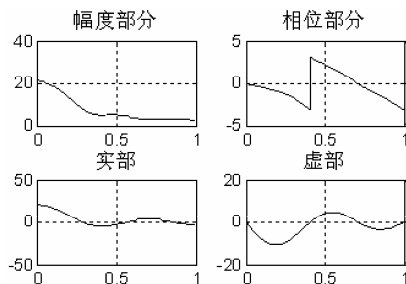


图 9-15 序列的离散时间傅里叶变换

【例 9-16】已知序列 $x(n) = (-0.45)^n$ ， $-5 \leq n \leq 5$ ，研究其周期性和共轭对称性。

显然该序列的离散时间傅里叶变换是以 2π 为周期的，因此为显示其周期性和其共轭对称性，对其离散时间傅里叶变换在 $[-2\pi, 2\pi]$ 之间选取 401 个频点，进行计算并绘图。

代码如下：

```
n=-5:5; %输入序列
x=(-0.45).^n;
k=-200:200; %在横坐标轴上分点
w=(pi/100)*k;
%进行 DTFT
X=x*(exp(-j*pi/100)).^(n*k);
magX=abs(X);
angX=angle(X);
```

```
%绘图
subplot(211);plot(w/pi,magX);
axis([-2 2 30 100]);
grid on;ylabel('|x|');title('幅度部分');
subplot(212);plot(w/pi,angX/pi);
axis([-2 2 -1 1]);
grid on;ylabel('弧度/pi');title('相位部分');
```

运行程序，效果如图 9-16 所示。

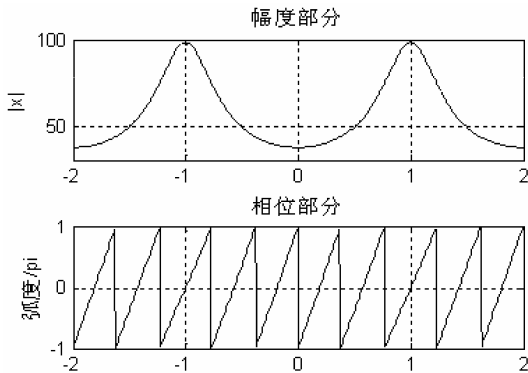


图 9-16 序列的离散时间傅里叶变换

从图 9-16 可以看出，序列的离散时间傅里叶变换是以 2π 为周期的，且当序列为实序列时，其离散傅里叶变换具有共轭对称性。

9.2.2 离散时间傅里叶变换的特性

本节讨论离散时间傅里叶变换在工程计算中的其他重要特性，如表 9-2 所示。以下设 $X(e^{j\omega})$ 为序列 $x(n)$ 的离散时间傅里叶变换。

表 9-2 离散时间傅里叶变换的特性

特 性	说 明
线性	$F[ax_1(n) + bx_2(n)] = aF[x_1(n)] + bF[x_2(n)]$
时移	$F[x(n-k)] = X(e^{j\omega})(e^{-j\omega k})$
频移	$F[x(n)e^{j\omega_0 n}] = X(e^{j(\omega-\omega_0)})$
共轭	$F[x^*(n)] = X^*(e^{-j\omega})$
反褶	$F[x(-n)] = X(e^{-j\omega})$
实序列的对称性	$x(n) = x_c(n) + x_o(n)$
卷积	$F[x_1(n) * x_2(n)] = F[x_1(n)]F[x_2(n)] = X_1(e^{j\omega})X_2(e^{j\omega})$
乘法	$F[x_1(n) \cdot x_2(n)] = F[x_1(n)] \otimes F[x_2(n)] = \frac{1}{2\pi} \int X_1(e^{j\theta})X_2(e^{j\omega})d\theta$
能量	$\varepsilon_x = \sum_{-\infty}^{\infty} x(n) ^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) ^2 d\omega = \int_0^{2\pi} \frac{ X(e^{j\omega}) ^2}{\pi} d\omega$

【例 9-17】要求验证 DTFT 的线性特性，即给出两个随机序列 $x_1(n)$ 和 $x_2(n)$ ($0 \leq n \leq 10$)，令其平均分布于 $[0, 1]$ 之间，然后用数值方法求离散时间傅里叶变换。

代码如下：

```
x1=rand(1,11);
x2=rand(1,11);
pha=3;
beta=2;
n=0:10;
k=0:500;
w=(pi/500)*k;
X1=x1*(exp(-j*pi/500)).^(n*k);
X2=x2*(exp(-j*pi/500)).^(n*k);
x=pha*x1+beta*x2;
X=x*(exp(-j*pi/500)).^(n*k);
X_ch=pha*X1+beta*X2;
error=max(abs(X-X_ch))
```

运行后得：

```
error = 7.3241e-015
```

从上面的计算结果可以看出，两个傅里叶变换数组之间的最大误差不超过 10^{-14} ，这在 MATLAB 的计算精度下可以认为是一样的。

【例 9-18】要求验证 DTFT 的采样移位性，即给出一个随机序列 $x_1(n)$ ($0 \leq n \leq 10$)，令其平均分布于 $[0, 1]$ 之间，并令 $y(n) = x(n-1)$ ，然后用数值方法求离散时间傅里叶变换。

代码如下：

```
x=rand(1,11);
n=0:10;
k=0:500;
w=(pi/500);
X=x*(exp(-j*pi/500)).^(n*k);
%信号移位三个样本点
y=x;
m=n+1;
Y=y*(exp(-j*pi/500)).^(m*k);
%校验
Y_ch=(exp(-j*1).^w).*X;
error=max(abs(Y-Y_ch))
运行得：
error = 3.2449
```

【例 9-19】为了验证 DTFT 的频移特性，这里采用图形的方法。两个序列为

$$x(n) = \cos(n\pi/2), \quad 0 \leq n \leq 10, \quad y(n) = e^{j\pi n/4} x(n)$$

代码如下：

```
n=0:100; %输入序列
x=cos(pi/100);
k=-100:100;
w=(pi/100)*k;
y=exp(j*pi*n/4).*x;
%进行 DTFT
X=x*(exp(-j*pi/100)).^(n*k);
```

```

Y=y*(exp(-j*pi/100)).^(n'*k);
%图形校验
subplot(221);plot(w/pi,abs(X));grid on;
axis([-1 1 0 60]);ylabel('|x|');title('X 的幅度');
subplot(222);plot(w/pi,angle(X)/pi);grid on;
axis([-1 1 -1 1]);ylabel('弧度/pi');title('X 的相位');
subplot(223);plot(w/pi,abs(Y));grid on;axis([-1 1 0 60]);
xlabel('以 pi 为单位的频率');ylabel('|x|');title('Y 的幅度');
subplot(224);plot(w/pi,angle(Y)/pi);grid on;axis([-1 1 -1 1]);
xlabel('以 pi 为单位的频率');ylabel('弧度/pi');title('Y 的相位');

```

运行程序，效果如图 9-17 所示。

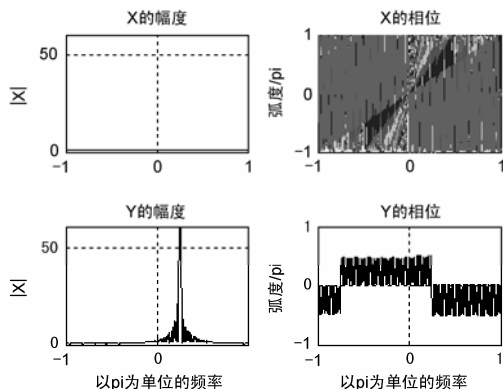


图 9-17 DTFT 的频移特性

9.3 数字滤波器的分析与实现

9.3.1 数字滤波器知识

数字滤波器与模拟滤波器相对应。在数字信号处理中，如果一个离散时间系统是用来对输入信号进行滤波处理，那么该系统就可以称为数字滤波器（Digital Filter）。因此，数字滤波器实际上就是离散时间系统。

数字滤波器的基本输入/输出关系为

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

其中 $h(n)$ 为滤波器的单位抽样响应。

数字滤波器的转移函数为

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n_b + 1)z^{-n_b}}{a(1) + a(2)z^{-1} + \cdots + a(n_a + 1)z^{-n_a}}$$

如果 $A(z) = 1$ ，则称滤波器为（有限冲激响应）FIR 滤波器；否则为（无限冲激响应）IIR 滤波器。令 $z = e^{j\omega}$ ，得到滤波器的频率响应为

$$H(e^{j\omega}) = k \frac{\prod_{r=1}^M (e^{j\omega} - q_r)}{\prod_{k=1}^N (e^{j\omega} - q_k)}$$

定义 $|H(e^{j\omega})|$ 为滤波器的幅频响应, 而 $\varphi(e^{j\omega})$ 为滤波器的相频响应。

9.3.2 数字滤波器的分析与实现

1. 数字滤波器的分析

(1) abs函数

abs 函数可以用来求信号的幅值, 其调用格式请参看以下示例。

【例 9-20】给定一个正弦信号, 求其傅里叶变换的幅值。

代码如下:

```
N=100;
n=0:N-1;
x=sin(2*pi*n/40);
X=fft(x); %对正弦信号 x 做 FFT
X=fftshift(X); %将频率为零的点移到频谱中央位置
X=abs(X); %求傅里叶变换的模值
plot(n,X);grid on;
title('信号傅里叶变换的幅值');
```

运行程序, 效果如图 9-18 所示。

(2) angle函数

angle 函数可以用来求信号的相角, 其调用格式请参看以下示例。

【例 9-21】给定一个信号, 求其傅里叶变换的相角。

代码如下:

```
N=100;n=0:N-1;
x=cos(2*pi*n/40)+cos(2*pi*n/20);
X=fft(x); %求信号傅里叶变换的相角
p=angle(X); %相位的解卷绕
p=unwrap(p);
plot(n,p);grid on;
title('信号傅里叶变换的相角图');
```

运行程序, 效果如图 9-19 所示。

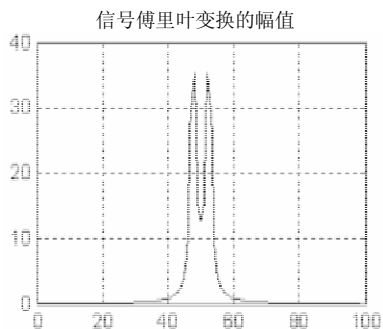


图 9-18 正弦信号傅里叶变换的幅值

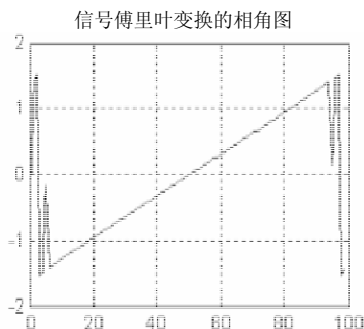


图 9-19 傅里叶变换的相角

(3) freqspace函数

freqspace 函数可以用来设置频率响应的频率间隔, 其调用格式参看帮助文档。

(4) freqs函数

freqs 函数可以用来求模拟滤波器的频率响应, 其调用格式参看以下示例。

【例 9-22】给定模拟滤波器的转移函数为 $H(s) = \frac{0.5s^2 + 2s + 1}{2s^2 + 0.2 + 1}$, 试绘制其频率响应。

代码如下:

```
den=[2 0.2 1];
num=[0.5 2 1];
freqs(num,den);
```

运行程序, 效果如图 9-20 所示。

(5) freqz函数

freqz 函数可以用来求数字滤波器的频率响应, 其调用格式参看示例。

【例 9-23】给定一个 FIR 滤波器的单位抽样响应为 $h(n)=\{1\ 2\ 3\ 4\ 5\ 6\}$, 绘制其幅频和相频响应。

代码如下:

```
a=1;
b=[1 2 3 4 5 6];
freqz(b,a); grid on;
```

运行程序, 效果如图 9-21 所示。

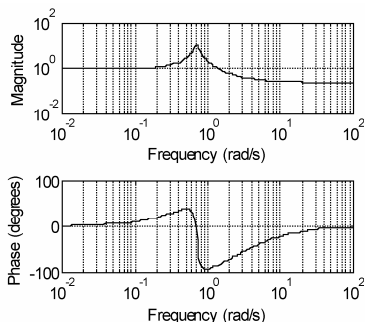


图 9-20 模拟滤波器的幅频响应和相频响应

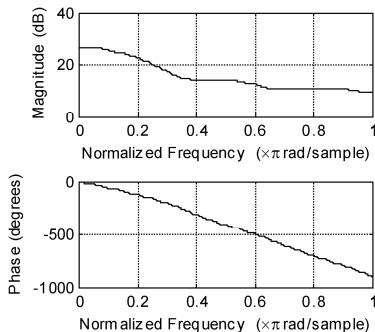


图 9-21 FIR 滤波器的幅频响应和相频响应

(6) impz函数

impz 函数可以用来求滤波器的单位冲击响应, 其调用格式参看示例。

【例 9-24】已知 IIR 滤波器的转移函数为 $H(z) = \frac{1 + 3z^{-1} + 2z^{-2}}{6 + z^{-1} - z^{-2}}$, 试绘制其单位冲激响应

波形。

代码如下:

```
num=[1 3 2];
den=[6 1 -1];
n=20;
impz(num,den,n); grid on;
```

运行程序，效果如图 9-22 所示。从结果中可以看出，系统是不稳定的。

(7) zplane 函数

zplane 函数可以用来绘制滤波器的零极点，其调用格式参看示例。

【例 9-25】绘制例 9-24 中滤波器的零极点图。

代码如下：

```
num=[1 3 2];
den=[6 1 -1];
zplane(num,den);
```

运行程序，效果如图 9-23 所示，从结果中可以看出，滤波器的两个极点均位于单位圆内，因而系统是稳定的。

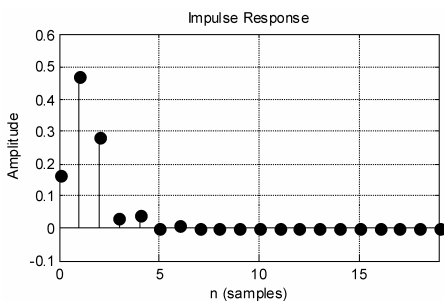


图 9-22 滤波器的单位冲激响应

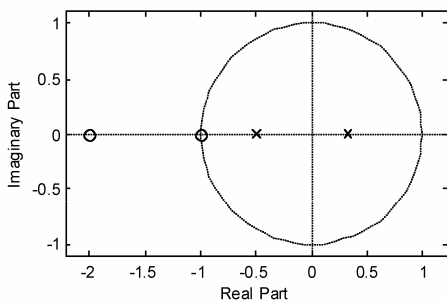


图 9-23 二阶系统零极点图

2. 滤波器的实现

(1) filter 函数

filter 函数可以用来实现直接 II 型滤波器，其调用格式参看示例。

【例 9-26】绘制 5 阶巴特沃兹低通滤波器的单位阶跃响应。

代码如下：

```
[m,n]=butter(5,0.2);
x=ones(1,50);
y=filter(m,n,x);
plot(y);grid on;
title('单位阶跃响应');
```

运行程序，效果如图 9-24 所示。

(2) filtic 函数

filtic 函数可以用来直接求 II 型滤波器的初始条件，其调用格式如下。

z=filtic(b,a,x,y): 给定滤波器的转移函数的分子分母多项式系数向量 **b** 和 **a** 及滤波器的输入信号 **x** 和输出信号 **y**，求对应的滤波器的初始条件 **z**。

(3) filtfilt 函数

filtfilt 函数可以用来对给定信号进行零相位滤波，其调用格式参看示例。

【例 9-27】对给定信号作零相位滤波。

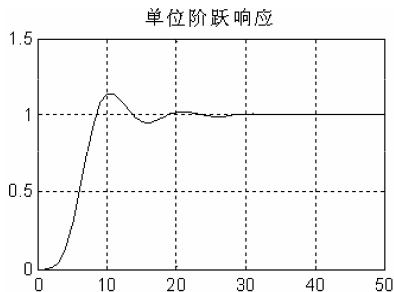


图 9-24 巴特沃兹低通滤波器的单位阶跃响应

代码如下：

```
N=100; %信号的长度
n=0:N-1;
x=sin(2*pi*n/40)+sin(2*pi*n/40); %待滤波信号
[a,b]=butter(5,0.2); %生成 5 阶巴特沃兹滤波器
y=filtfilt(a,b,x); %零相位滤波
subplot(211);stem(n,x);
xlabel('n');ylabel('x(n)');grid on;
subplot(212);stem(n,y);axis([0 100 -2 2]);
xlabel('n');ylabel('y(n)');grid on;
```

运行程序，效果如图 9-25 所示。从图中可以看出，滤波后的信号相对于原来信号的相位延迟为 0。

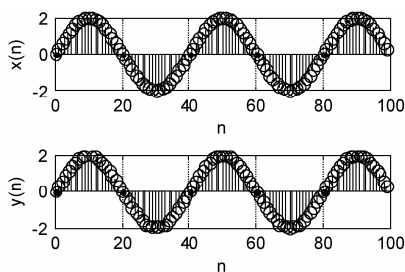


图 9-25 零相位滤波

(4) fftfilt 函数

fftfilt 函数可以用来进行基于 FFT 的 FIR 滤波，其调用格式参看示例。

【例 9-28】信号 $x(n)$ 为正弦信号加上均值为 0，功率为 0.1 的白噪声。其长度为 9999 点。利用 fftfilt 函数对其进行滤波处理。

代码如下：

```
p=0.1; %白噪声的功率
N=999; %数据长度
n=0:N-1;
x1=rand(1,N); %得到均值为 0，功率为 0.1 的白噪声
x1=(x1-mean(x1))*sqrt(p/var(x1)); %调整白噪声的均值和功率
x2=cos(2*pi*n/85);
x=x1+x2; %叠加，得到待滤波信号
b=fir1(12,0.2); %设计 12 阶 FIR 滤波器
y=fftfilt(b,x); %基于 FFT 的 FIR 滤波
subplot(211);plot(x);
xlabel('n');ylabel('x(n)');grid on;
subplot(212);plot(y);
xlabel('n');ylabel('y(n)');grid on;
```

运行程序，效果如图 9-26 所示。

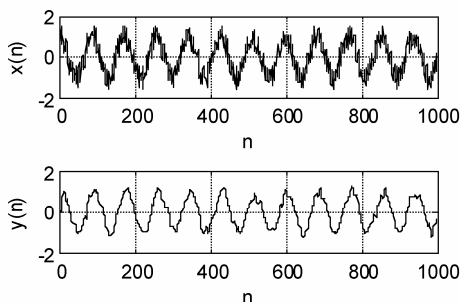


图 9-26 基于 FFT 的 FIR 滤波

9.4 IIR 数字滤波器的设计法

9.4.1 冲激响应不变法

冲激响应不变法设计 IIR 数字滤波器的基本原理是：对具有传递函数 $H_a(s)$ 的模拟滤波器的冲激响应 $h_a(t)$ ，将以周期 T 采样所得的离散序列 $h_a(nT)$ ，作为数字滤波器的单位冲激响应 $h(n)$ ，即 $h(n)$ 与 $h_a(t)$ 满足如下关系

$$h(n) = h_a(t) \big|_{t=nT}$$

当模拟滤波器的系统函数 $H_a(s)$ 只有单阶极点时，则利用冲激响应不变法所得数字滤波器的系统函数 $H(z)$ 有以下对应关系

$$H_a(s) = \sum_{k=1}^N \frac{A_k}{s - s_k} \rightarrow H(z) = \sum_{k=1}^N \frac{A_k}{1 - e^{s_k T} z^{-1}}$$

在 MATLAB 工具箱中，提供了专用函数 `impinvar` 来实现以上计算，其调用格式参看示例。

【例 9-29】一个 4 阶 Butterworth 低通滤波器的系统函数为：

$$H_a(s) = \frac{1}{s^4 + \sqrt{5}s^3 + 2s^2 + \sqrt{2}s + 1}$$

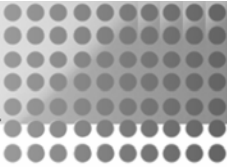
试用冲激响应不变法求出 Butterworth 数字滤波器的系统函数。

代码如下：

```
>> num1=[1]; %模拟滤波器系统函数的分子
den1=[1,sqrt(5),2,sqrt(2),1]; %模拟滤波器系统函数的分母
[num,den]=impinvar(num1,den1) %求数字低通滤波器的系统函数
```

运行结果为：

```
num =
    -0.0000    0.0942    0.2158    0.0311
den =
    1.0000   -2.0032    1.9982   -0.7612    0.1069
```



所以，IIR 数字滤波器的传递函数为

$$H(z) = \frac{0.00942z^{-1} + 0.2158z^{-2} + 0.0311z^{-3}}{1 - 2.0032z^{-1} + 1.99826z^{-2} - 0.10697612z^{-3} + 0.1069z^{-4}}$$

若再利用 `dimpulse` 函数就可以求出数字滤波器的单位冲激响应。

9.4.2 双线性变换法

MATLAB 信号处理工具箱为实现双线性变换提供了函数 `bilinear`，其调用格式参看示例。

【例 9-30】一个 3 阶模拟 Butterworth 低通滤波器的传递函数为

$$H(s) = \frac{1}{s^3 + \sqrt{3}s^2 + \sqrt{2}s + 1}$$

试用双线性变换法求出数字 Butterworth 低通滤波器的传递函数。

代码如下：

```
num1=[1]; %模拟滤波器系统函数的分子
den1=[1,sqrt(3),sqrt(2),1]; %模拟滤波器系统函数的分母
[num,den]=bilinear(num1,den1,1) %求数字滤波器的传递函数
```

运行结果为：

```
num =
    0.0533    0.1599    0.1599    0.0533
den =
    1.0000   -1.3382    0.9193   -0.1546
```

9.4.3 IIR 数字滤波器的频率变换设计法

IIR 数字滤波器的频率变换设计法的基本思想是，由原型低通滤波器，通过频率变换形成不同形式的数字滤波器。其实现过程分为两种：第一种，首先根据滤波器设计要求，设计模拟原型低通滤波器，然后进行频率变换，将其转换为相应的模拟滤波器（高通、带通等），最后利用冲激响应不变法或双线性变换法，将模拟滤波器数字化成相应的数字滤波器；第二种，首先根据滤波器设计要求，设计模拟原型低通滤波器，然后利用冲激响应不变法或双线性变换法，将模拟原型低通滤波器数字化成数字原型低通滤波器，最后进行频率变换，将数字原型低通滤波器转换为相应的数字滤波器（高通、带通等）。

1. MATLAB 的典型设计

为了实现频率变换设计 IIR 数字滤波器，MATLAB 在信号处理工具箱中提供了大量的 IIR 数字滤波器设计的相关函数，其中包括 IIR 滤波器阶次估计函数（见表 9-3）、模拟低通滤波器原型设计函数（见表 9-4），以及模拟滤波器变换函数（见表 9-5）。每个函数有多种调用方法，读者可以通过帮助文档来了解。

表 9-3 IIR 滤波器阶次估计函数

函 数 名	功 能
buttord	计算 Butterworth 滤波器的阶次和截止频率



续表

函 数 名	功 能
cheb1ord	计算 Chebyshev I 型滤波器的阶次
cheb2ord	计算 Chebyshev II 型滤波器的阶次
ellipord	计算椭圆滤波器最小阶次

表 9-4 模拟低通滤波器原型设计函数

函 数 名	功 能
besselap	Bessel 模拟低通滤波器原型设计
buttap	Butterworth 模拟低通滤波器原型设计
cheb1ap	Chebyshev I 型模拟低通滤波器原型设计
cheb2ap	Chebyshev II 型模拟低通滤波器原型设计
ellipap	椭圆模拟低通滤波器原型设计

表 9-5 模拟滤波器变换函数

函 数 名	功 能
lp2bp	把低通模拟滤波器转换成带通滤波器
lp2bs	把低通模拟滤波器转换成带阻滤波器
lp2hp	把低通模拟滤波器转换成高通滤波器
lp2	改变低通模拟滤波器的截止频率

因此,利用这些函数,IIR 数字滤波器的频率变换设计将变得非常简单。利用 MATLAB 设计 IIR 数字滤波器可分以下几步来实现。

- ① 按一定规则将数字滤波器的技术指标转换为模拟低通滤波器的技术指标[a]。
- ② 根据转换后的技术指标使用滤波器阶数函数,确定滤波器的最小阶次数 N 和截止频率 ω_c 。
- ③ 利用最小阶数 N 产生模拟低通滤波器原型。
- ④ 利用截止频率 ω_c 把模拟低通滤波器原型转换成模拟低通、高通、带通或带阻滤波器。
- ⑤ 利用冲激响应不变法或双线性变换法把模拟滤波器转换成数字滤波器。

下面以巴特沃思(Butterworth)滤波器设计函数为例,介绍实现步骤。其他类型的滤波器设计函数用法可类推。

利用 buttord 函数求模拟滤波器最小阶数和截止频率。其调用格式参看示例。

【例 9-31】设计一个数字信号处理系统,它的采样率为 100Hz,希望在该系统中设计一个 Butterworth 型高通数字滤波器,使其通带中允许的最大衰减为 0.5dB,阻带内的最小衰减为 40dB,通带上限临界频率为 30Hz,阻带下限临界频率为 40Hz。

代码如下:

```
%把数字滤波器频率特征转换成模拟滤波器的频率特征
wp=30*2*pi;ws=40*2*pi;
rp=0.5;rs=40;Fs=100;
```

```

[N,Wc]=buttord(wp,ws,rp,rs,'s'); %选择滤波器的最小阶数
[Z,P,K]=butter(N); %创建 Butterworth 低通滤波器原型
[A,B,C,D]=zp2ss(Z,P,K); %零极点增益模型转换为状态空间模型
[AT,BT,CT,DT]=lp2hp(A,B,C,D,Wc); %实现低通向高通的转变
[num,den]=ss2tf(AT,BT,CT,DT); %状态空间模型转换为传递函数模型
%运用双线性变换法把模拟滤波器转换为数字滤波器

[num1,den1]=bilinear(num,den,Fs);
[H,W]=freqz(num1,den1); %求频率响应
plot(W*Fs/(2*pi),abs(H));grid on; %绘制频率响应曲线
xlabel('频率/Hz');ylabel('幅值');

```

运行程序，效果如图 9-27 所示。

【例 9-32】利用冲激响应不变法设计一个低通 Chebyshev I 型数字滤波器，其通带上限临界频率为 0.3Hz，阻带临界频率为 0.4Hz，采样频率为 1000Hz，在通带内的最大衰减为 0.3dB，阻带内的最小衰减为 80dB。

代码如下：

```

%把数字滤波器频率特征转换成模拟滤波器的频率特征
wp=300*2*pi;ws=400*2*pi;
rp=0.3;rs=80;Fs=1000;
[N,Wc]=cheblord(wp,ws,rp,rs,'s'); %选择滤波器的最小阶数
[Z,P,K]=cheblap(N,rp); %创建 Butterworth 低通滤波器原型
[A,B,C,D]=zp2ss(Z,P,K); %零极点增益模型转换为状态空间模型
[AT,BT,CT,DT]=lp2lp(A,B,C,D,Wc); %实现低通向高通的转变
[num,den]=ss2tf(AT,BT,CT,DT); %状态空间模型转换为传递函数模型
%运用冲激响应不变法把模拟滤波器转换为数字滤波器

[num1,den1]=impinvar(num,den,Fs);
[H,W]=freqz(num1,den1); %求频率响应
plot(W*Fs/(2*pi),abs(H));grid on; %绘制频率响应曲线
xlabel('频率/Hz');ylabel('幅值');

```

运行程序，效果如图 9-28 所示。

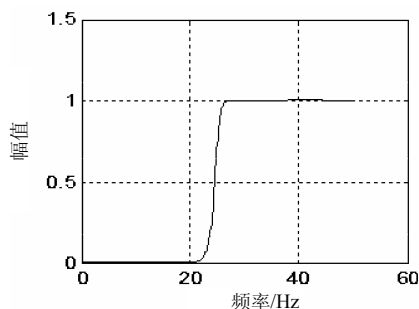


图 9-27 Butterworth 高通滤波器频率响应

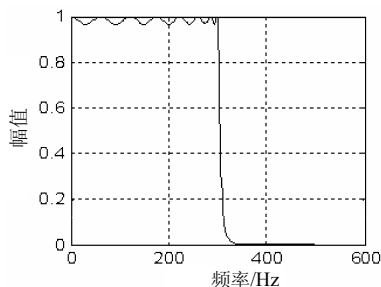


图 9-28 Chebyshev I 型数字滤波器频率响应

2. MATLAB的直接设计

另外，MATLAB 信号处理工具箱还提供了几个直接设计 IIR 数字滤波器的函数，如表 9-6 所示。

表 9-6 IIR 数字滤波器设计函数

函 数 名	功 能
butter	巴特沃思(ButterWorth)模拟和数字滤波器设计
cheby1	契比雪夫(Chebyshev) I 型滤波器设计(通带波纹)
cheby2	契比雪夫(Chebyshev) II 型滤波器设计(阻带波纹)
ellip	椭圆(Cauer)滤波器设计
maxflat	一般巴特沃思数字滤波器设计(最平滤波器)
prony	利用 Prony 法进行时域 IIR 滤波器设计
stmcb	利用 Steiglitz-McBride 迭代法求线性模型
yulewalk	递归数字滤波器设计

下面仍以巴特沃思滤波器设计函数 `butter` 为例进行说明, 其他函数的使用类似。直接设计 IIR 数字滤波器的步骤如下。

- ① 利用 `buttord` 函数求数字滤波器的最小阶数和截止频率。
- ② 利用函数 `butter` 直接设计。

由此可见, 在通常情况下, 有了 IIR 滤波器阶次估计, 利用直接设计 IIR 数字滤波器的函数, 数字滤波器设计问题也就解决了。

值得一提的是:

① 在直接设计 IIR 数字滤波器的函数中, 采用的是双线性变换函数 `bilinear`, 如果要用冲激响应不变法就需要分步进行, 即采用典型设计法。

② 表 9-6 中的 `butter` 函数、`cheby1` 函数、`cheby2` 函数和 `ellip` 函数, 不仅可以设计数字滤波器, 而且还可以设计模拟滤波器。例如, 当采用 `butter(n, wn, 's')`、`butter(n, wn, 'high', 's')` 和 `butter(n, wn, 'stop', 's')` 时, 用于设计模拟滤波器, 此时截止频率的单位为 rad/s, 它可以大于 1.0。

【例 9-33】试设计一个阻带 IIR 数字滤波器, 具体要求是, 通带截止频率 $\omega_{p1}=600\text{Hz}$, $\omega_{p2}=800\text{Hz}$; 阻带截止频率 $\omega_{s1}=700\text{Hz}$, $\omega_{s2}=750\text{Hz}$; 通带内的最大衰减为 $r_p=0.1\text{dB}$; 阻带内的最小衰减为 $r_s=45\text{dB}$; 采样频率为 $F_s=2000\text{Hz}$ 。

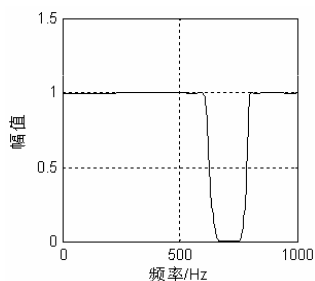


图 9-29 椭圆带阻滤波器的频率响应

代码如下:

```
wp1=600;wp2=800;ws1=700;
ws2=750;rp=0.1;rs=50;Fs=2000;
wp=[wp1,wp2]/(Fs/2);ws=[ws1,ws2]/(Fs/2);
%利用 Nyquist 频率进行归一化
[N,wc]=ellipord(wp,ws,rp,rs,'z'); %求滤波器阶数
[num,den]=ellip(N,rp,rs,wc,'stop'); %求滤波器传递函数
[H,W]=freqz(num,den); %绘制频响应曲线
plot(W*Fs/(2*pi),abs(H));grid on;
xlabel('频率/Hz');ylabel('幅值');
```

运行程序, 效果如图 9-29 所示。

【例 9-34】设计一个 Chebyshev I 型带通滤波器, 具体要求: $\omega_{p1}=55\text{Hz}$, $\omega_{p2}=75\text{Hz}$, $\omega_{s1}=50\text{Hz}$, $\omega_{s2}=80\text{Hz}$, $r_p=0.5$, $r_s=55$, $F_s=200\text{Hz}$ 。

代码如下:

```
wp1=55;wp2=75;ws1=50;
```

```

ws2=80;rp=0.5;rs=55;Fs=200;
wp=[wp1,wp2]/(Fs/2);ws=[ws1,ws2]/(Fs/2);
%利用 Nyquist 频率进行归一化
[N,wc]=cheb1ord(wp,ws,rp,rs); %求滤波器阶数
[num,den]=cheby1(N,rp,wc); %求滤波器传递函数
[H,W]=freqz(num,den); %绘制频响应曲线
plot(W*Fs/(2*pi),abs(H));grid on;
xlabel('频率/Hz');ylabel('幅值');
运行程序,效果如图 9-30 所示。

```

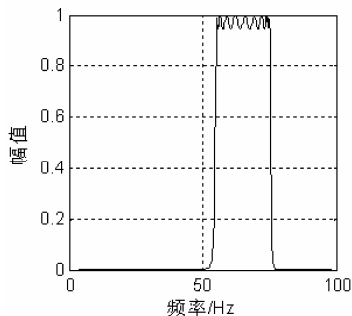


图 9-30 Chebyshev I 型带通滤波器的频率响应

9.5 FIR 数字滤波器设计法

9.5.1 窗函数设计法

窗函数设计法的基本原理是：为了用 $H(e^{j\omega}) = \sum_{n=0}^{M-1} h(n)e^{-jn\omega}$ 逼近理想的频率响应

$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d(n)e^{-jn\omega}$ ，获取有限长序列 $h(n)$ 的最有效方法是用一个有限长的窗口函数 $\omega(n)$ 来截取无限长序列 $h_d(n)$ ，即

$$h(n) = \omega(n)h_d(n)$$

$$H_d(e^{j\omega}) = \begin{cases} e^{j\alpha\omega}, & \omega_{c1} \leq |\omega| \leq \omega_{c2} \\ 0, & 0 < |\omega| < \omega_{c1}, \omega_{c2} < |\omega| < \pi \end{cases}$$

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\omega}) e^{jn\omega} d\omega = \frac{\sin[\omega_c(n-\alpha)]}{\pi(n-\alpha)}$$

在 MATLAB 信号处理工具箱中为用户提供了 Boxcar (矩形)、Bartlet (巴特利特)、Hanning (汉宁) 等窗函数，这些窗函数可通过 “help signal\signal” 获取。由于这些窗函数的调用格式相同，下面仅以 Boxcar (矩形) 函数为例说明其调用格式。

格式： w=boxcar(M)

功能： 返回 M 点矩形空序列。

窗的长度又称为窗函数设计 FIR 数字滤波器的阶数。根据卷积理论可知， $H(e^{j\omega})$ 是理想的频率响应与窗函数频率响应的圆周卷积。

$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta}) W(e^{j(\omega-\theta)}) d\theta$$

因此， $H(e^{j\omega})$ 逼近程度的好坏完全取决于窗函数的频率特性。表 9-7 给出了部分窗函数的频率特性。

表 9-7 在相同条件下部分窗函数的频率特性

名 称	近似过渡带宽	过渡带宽	最小阻带衰减
Boxcar(矩形)	$4\pi/M$	$1.8\pi/M$	21dB
Bartlet(巴特利特)	$8\pi/M$	$4.2\pi/M$	25dB
Hanning(汉宁)	$8\pi/M$	$6.2\pi/M$	44dB
Hamming(哈明)	$8\pi/M$	$6.6\pi/M$	51dB
Blackman(布莱克曼)	$12\pi/M$	$11\pi/M$	74dB

【例 9-35】用矩形窗设计线性相位 FIR 低通滤波器，该滤波器的通带截止频率 $\omega_c = \pi/4$ ，单位脉冲 $h(n)$ 的长度 $M=21$ ，绘出 $h(n)$ 及其幅度响应特性曲线。

代码如下：

```

M=21;wc=pi/4;                %理想低通滤波器参数
n=0:M-1;r=(M-1)/2;
nr=n-r+eps*((n-r)==0);
hdn=sin(wc*nr)/pi./nr;        %计算理想低通单位脉冲响应 hdn
if rem(M,2)~=0
    hdn(r+1)=wc/pi;           %M 为奇数时，处理 n=r 点的 0/0 型
end;
wn1=boxcar(M);                %矩形窗
hn1=hdn.*wn1';                %加窗
subplot(211);stem(n,hn1,'-o');line([0,20],[0,0]);
xlabel('n');ylabel('h(n)');title('矩形窗设计的 h(n)');
hw1=fft(hn1,512);w1=2*[0:511]/512; %求频谱
subplot(212);plot(w1,20*log10(abs(hw1)));
xlabel('w/pi');ylabel('幅度(dB)');title('幅度特性(dB)');

```

运行程序，效果如图 9-31 所示。

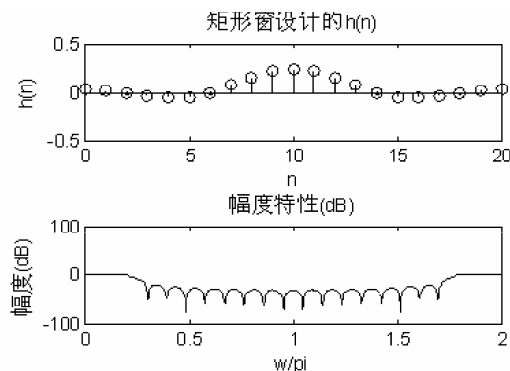


图 9-31 矩形窗设计 FIR 滤波器

在 MATLAB 信号处理工具箱中，除提供窗函数命令外，还提供用窗函数法设计 FIR 数字滤波器的专用命令 `fir1`。利用该函数可设计出具有标准频率响应的 FIR 滤波器，所得滤波器系数（单位冲激响应）为实数。其基本调用格式参看示例。

【例 9-36】设计一个 24 阶 FIR 带通滤波器，通带为 $[0.3\ 0.7]$ 。

代码如下：

```

wc=[0.3 0.7]; %设置阻带的范围
b=fir1(24,wc); %调用 fir1 函数
freqz(b); %绘制滤波器的频率响应曲线
figure(2);
stem(b,''); %绘制单位冲激响应序列
line([0 25],[0 0]);
xlabel('n'); ylabel('h(n)');

```

运行程序，效果如图 9-32 所示。

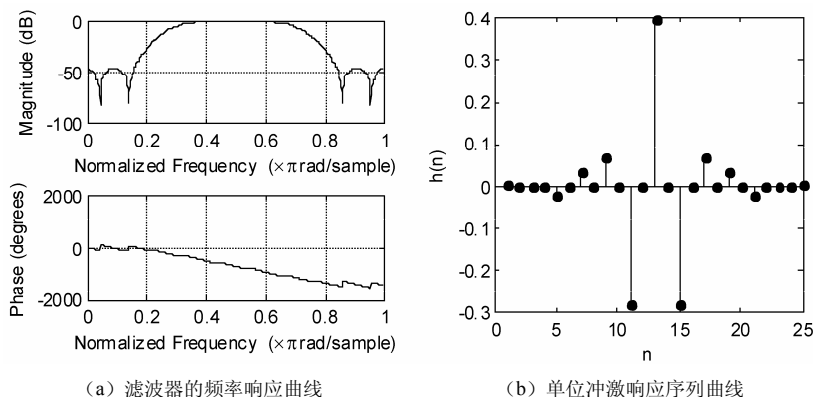


图 9-32 带通滤波器的频率响应及单位冲激响应

【例 9-37】用窗函数设计一个多带通滤波器，归一化的通带是 $[0.2, 0.4]$ ， $[0.6, 0.8]$ ， $[1.0, 1.2]$ 。注意高频端为通带，故滤波器的阶数应为偶数。这里取 $N=40$ 。

代码如下：

```

wc=[0.2 0.4 0.6 0.8]; %设置阻带的范围
b=fir1(40,wc,'dc-1'); %调用 fir1 函数
freqz(b); %绘制滤波器的频率响应曲线
figure(2);
stem(b,''); %绘制单位冲激响应序列
line([0 45],[0 0]);
xlabel('n'); ylabel('h(n)');

```

运行程序，效果如图 9-33 所示。

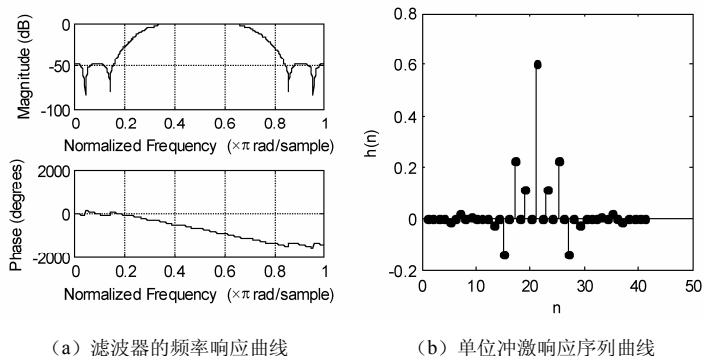


图 9-33 多带通滤波器的频率响应及单位冲激响应

9.5.2 频率抽样法

频率抽样法设计的基本原理：对所期望的滤波器的频率响应 $H_d(e^{j2\pi k/N})$ ，在频域上进行采样，以此来确定 FIR 滤波器，即令

$$H(k) = H_d(e^{j2\pi k/N})$$

对于线性相位 FIR 滤波器 $H(k)$ ，在设计时还应满足采样值的幅度与相位约束条件。

由频率抽样法设计的滤波器，在每个采样点上，频率响应将严格与理想特性一致，而在采样点之间的频率响应，则是由各采样点的内插函数延伸叠加来形成，因此，如果各采样点之间的理想特性越平缓，则内插值就越接近理想值，逼近也就越好。

在频域采样时，理想的矩形特性经过采样，在通带的边缘由于采样点之间的突然变化引起了频率特性的起伏变化，使阻带衰减减小。在窗函数法中用加宽过渡带来换取阻带的衰减，在这里可以通过选择适当的通带与阻带之间的过渡带的频率采样来达到峰值逼近误差最小。过渡带的频率采样值以 H_T 表示， H_T 究竟取几点，这要根据实际需要来确定，一般 H_T 取 1~3 点以后就能得到良好的效果。

由于受 N 个采样点的限制，滤波器的截止频率 ω_c 不能任意选择，为了能自由选择 ω_c ，应增加采样点数 N 。

【例 9-38】用频率采样法设计一个具有线性相位的低通滤波器，其理想频率选择性为

$$|H(e^{j\omega})| = \begin{cases} 1, & 0 \leq \omega \leq \omega_c \\ 0, & \text{其他} \end{cases}$$

已知截止频率为 0.5π ，抽样点数 $N=33$ 。

由于采样的 $|H(k)|$ 关于 $\omega=\pi$ 对称，抽样点数 $N=33$ ，采样点之间的频率间隔为 $2\pi/33$ ，截止频率为 0.5π ，因此，截止频率抽样点的位置应为 $0.5 \times 33/2 = 8.25 \approx 8$ 。所以，在 $0 \leq \omega \leq \pi$ 区域，抽样的 $H(k)$ 的幅度满足

$$|H(k)| = \begin{cases} 1, & 0 \leq k \leq \text{Int}[N\omega_c/2\pi] = (N-1)/4 \\ 0, & \text{Int}[N\omega_c/2\pi]+1 \leq k \leq (N-1)/2 \end{cases} = \begin{cases} 1, & 0 \leq k \leq 8 \\ 0, & 9 \leq k \leq 16 \end{cases}$$

若设计滤波器的相位满足

$$\theta(k) = \begin{cases} -\pi k(N-1)/N, & k=0, \dots, (N-1)/2 \\ \pi(N-k)(N-1)/N, & k=(N+1)/2, \dots, N-1 \end{cases}$$

所以，抽样 $H(k)$ 应满足

$$H(k) = \begin{cases} |H(k)| e^{\frac{j\pi k(N-1)}{N}}, & k=0, \dots, (N-1)/2 \\ |H(k)| e^{\frac{j\pi(N-1)(N-k)}{N}}, & k=(N+1)/2, \dots, N-1 \end{cases}$$

代码如下:

```
clear;N=33;
H=[ones(1,9),zeros(1,15),ones(1,9)]; %确定抽样点的幅度大小
%H(1,10)=0.5;H(1,24)=0.5; %设置过渡点
k=0:(N-1)/2;k1=(N+1)/2:(N-1);
A=[exp(-j*pi*k*(N-1)/N),exp(j*pi*k1*(N-1)/N)]; %确定抽样点的相位大小
HK=H.*A; %求抽样点的 H(k)
hn=ifft(HK); %求出 FIR 的单位冲激响应 h(n)
freqz(hn,1,256); %画幅频相频曲线
figure(2);
stem(real(hn),'.'); %绘制单位冲激响应的实部
line([0 35],[0 0]);
xlabel('n');ylabel('h(n)');
```

运行程序, 效果如图 9-34 所示。

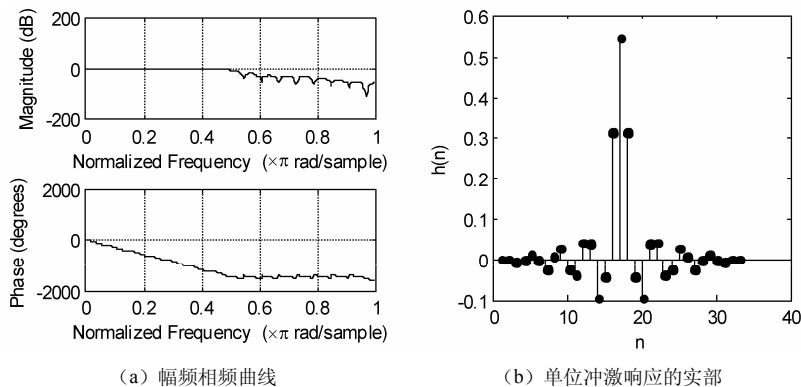


图 9-34 FIR 滤波器频率特性曲线

为了增加阻带衰减, 可以加宽过渡带, 如增加一个过渡点 $H(k) = 0.5$, 在程序中增加一条命令:

```
H(1,10)=0.5;H(1,24)=0.5; %设置过渡点
```

运行程序, 效果如图 9-35 所示, 从图中可以看出最小阻带衰减约 -40dB, 代价是增加了过渡带宽度。注意: 在该例子中, 所获得的滤波器系数为复数。

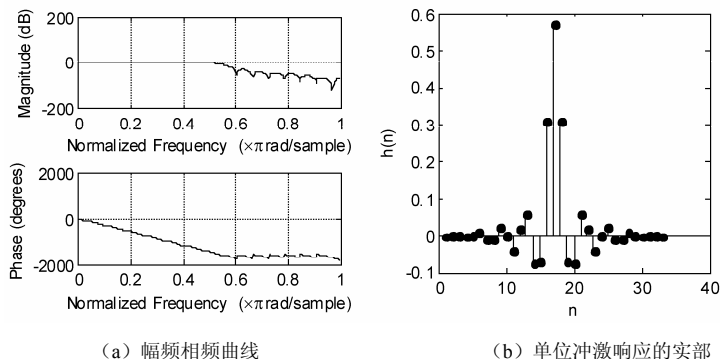


图 9-35 增加阻带的 FIR 滤波器频率特性曲线

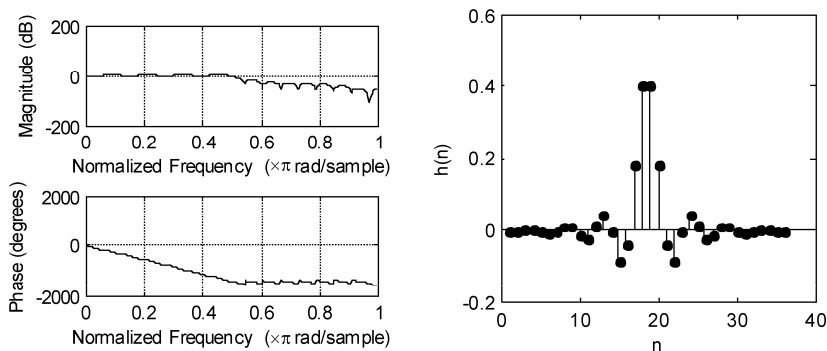
在 MATLAB 信号处理工具箱中，为频率抽样法设计 FIR 滤波器提供了专用函数命令 `fir2`。该函数的功能是，利用频率抽样法，设计任意响应的 FIR 数字滤波器，所得滤波器系数为实数，具有线性相位，满足偶对称性 $B(k)=B(N+2-k)$ ， $k=1,2,\dots,N+1$ 。其调用格式参看示例。

【例 9-39】试用频率抽样法设计一个 FIR 低通滤波器，该滤波器的截止频率为 0.5π ，频率抽样点数为 35。

代码如下：

```
clear; N=35;
F=[0:1/35:1]; %设置抽样点的频率，抽样频率必须含 0 和 1
A=[ones(1,16),zeros(1,N-15)]; %设置抽样点相应的幅值
B=fir2(N,F,A);
freqz(B); %绘制滤波器的频率响应曲线
figure(2);stem(B,'.'); %绘制单位冲激响应的实部
line([0 35],[0 0]);
xlabel('n');ylabel('h(n)');
```

运行程序，效果如图 9-36 所示。



(a) 频率响应曲线

(b) 单位冲激响应序列曲线

图 9-36 滤波器的频率响应和单位冲激响应序列

【例 9-40】试用 `fir2` 设计一个 40 阶的多带 FIR 滤波器，其频率特性为

$$|H(e^{j\omega})| = \begin{cases} 1, & 0 \leq \omega \leq 0.4\pi \\ 0, & 0.4\pi < \omega \leq 0.6\pi \\ 0.5, & 0.6\pi < \omega \leq 0.7\pi \\ 0, & 0.7\pi < \omega \leq 0.8\pi \\ 1, & 0.8\pi < \omega \leq \pi \end{cases}$$

代码如下：

```
f=0:0.002:1; %设置抽样点的频率
m(1:201)=1;m(202:301)=0;m(301:351)=0.5; %设置抽样点相应的幅值
m(352:401)=0;m(402:501)=1;
plot(f,m,'r');hold on;
b=fir2(40,f,m);
[h,f1]=freqz(b);
f1=f1./pi;
plot(f1,abs(h)); %频率归一化
legend('理想滤波器','设计滤波器');
xlabel('归一化频率');ylabel('幅值');
```

运行程序，效果如图 9-37 所示。

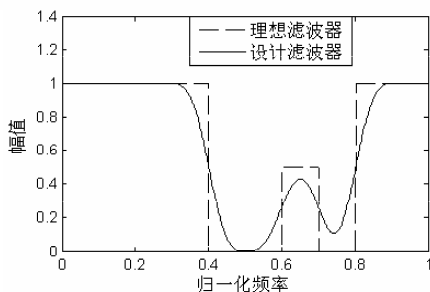


图 9-37 理想滤波器与设计滤波器对比图

9.5.3 MATLAB的其他相关函数

在 MATLAB 信号处理工具箱中, 不仅提供了 FIR 窗函数设计法和频率抽样设计法专用命令, 同时还提供了等波纹最佳一致逼近法、最小二乘逼近法等命令函数, 这为用户的 FIR 数字滤波器的设计提供了方便。

1. fircls函数

fircls 函数为最小二乘逼近法的线性相位 FIR 滤波器。其调用格式参看示例。

【例 9-41】试用 fircls 设计一个带通滤波器, 其通带为 $[0.4, 0.8]$, 允许波动的范围为 $-0.1 \sim 0.3$ 。用所设计的滤波器对信号

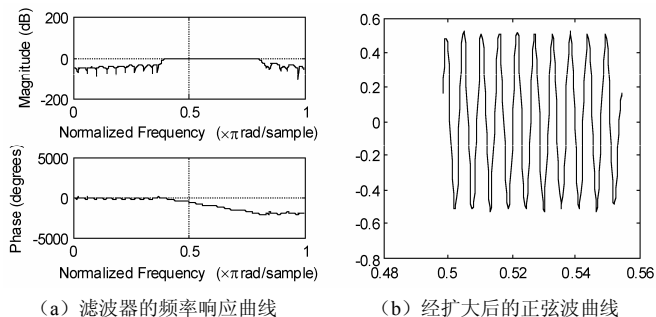
$$\sin(2\pi \times 30t) + 0.5 \sin(2\pi \times 180t) + 0.2 \sin(2\pi \times 270t)$$

滤波 (信号采样频率为 600Hz)。

代码如下:

```
clear;N=51; %所设计滤波器的阶数
%在  $0 < w < 0.4$ , 频率响应幅度为 0;  $0.9 < w < 1$ , 频率响应幅度为 1
f=[0 0.4 0.8 1];a=[0 1 0];
u=[0.02 2.02 0.61]; %抽样点允许波动的上限幅度
lo=[-0.02 0.121 -0.01]; %抽样点允许波动的下限幅度
b=fircls(N,f,a,u,lo);
freqz(b); %绘制滤波器的频率响应
t=0:1/600:1;
sig=cos(2*pi*30*t)+0.5*sin(2*pi*180*t)+0.2*sin(2*pi*270*t);
%信号由 30,180,270Hz 的正弦波叠加而成, 归一化频率分别为 0.1, 0.6, 0.9
newsig=fftfilt(b,sig); %用重叠相加法 FFT 实现对信号向量快速滤波
ft=t(300:333);ns=newsig(300:333); %取一段, 放大显示 (时间轴), 放大倍数为 12
zns=interp(ns,8);znt=interp(ft,8); %插值 (上采样)
figure(2);plot(znt,zns); %注意显示的结果为一幅值约为 0.6 的正弦波
```

运行程序, 效果如图 9-38 所示。



(a) 滤波器的频率响应曲线

(b) 经扩大后的正弦波曲线

图 9-38 带通滤波器的频率响应及信号滤波后的结果

2. fircls1 函数

fircls1 函数为有限制条件的最小二乘逼近法的低通和高通 FIR 数字滤波器。其调用格式参看示例。

【例 9-42】试用 fircls1 设计低通 FIR 数字滤波器，归一化截止频率为 0.3，通带波纹为 0.03，阻带波纹为 0.009。

代码如下：

```
clear;n=56; %滤波器的阶数
wo=0.32; %截止频率
dp=0.03; %通带波纹
ds=0.009; %阻带波纹
h=fircls1(n,wo,dp,ds);
[H,f]=freqz(h);
plot(f/pi,abs(H)); grid on; %绘制滤波器的频率特性曲线
xlabel('归一化频率');ylabel('幅度');
wp=0.3; %通带频率
ws=0.32; %截止频率
k=9; %通带加权
h1=fircls1(n,wo,dp,ds,wp,ws,k);
[H1,f1]=freqz(h1);
figure(2);
plot(f1/pi,abs(H));grid on;
xlabel('归一化频率');ylabel('幅度');
```

运行程序，效果如图 9-39 所示。

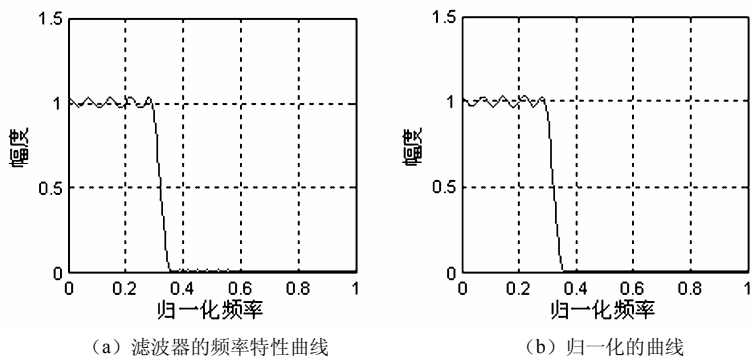


图 9-39 用 fircls1 设计的滤波器的频率特性

3. firls 函数

firls 函数为最小二乘逼近法的线性相位 FIR 数字滤波器。其调用格式参看示例。

【例 9-43】设计一个特殊滤波器，使其频率响应的频带在[0,0.4]内从 1.0 线性降到 0.5，而频带在[0.7, 0.9]内恒为 1.0，其他频带不予考虑。

代码如下：

```
b=firls(30,[0,0.4,0.7,0.9],[1.0,0.5,1.0,1.0]);
freqz(b)
```

运行程序，效果如图 9-40 所示。

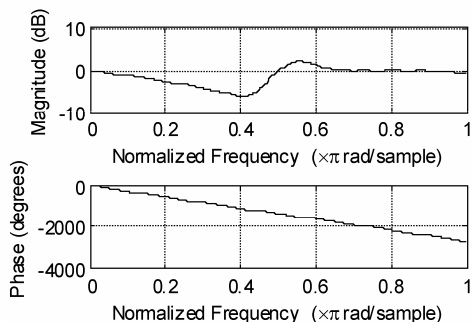


图 9-40 用 firls 设计的滤波器的频率响应

4. firrcos函数

firrcos 函数为升余弦 FIR 滤波器设计函数。

格式: B=firrcos(N, Fc, DF, Fs)

功能: 返回一个 N 阶低通线性且具有升余弦过渡频带的 FIR 滤波器。Fc 为截止频率, Fs 为抽样频率, DF 为过渡带, 三者之间必须满足滤波足够小以使 $F_c \pm DF/2$ 在 $[0, Fs/2]$ 范围。默认时, 采样频率 $F_s=2$ 。

5. remez函数

remez 函数为 Parks-McClellan 优化等波纹 FIR 滤波器设计函数。与其他设计法相比, 采用 remez 算法实现线性相位 FIR 数字滤波器的等波纹最佳一致逼近设计, 其优点是在设计指标相同时, 可使滤波器阶数最低; 在阶数相同时, 使通带最平坦, 阻带最小衰减最大; 通带和阻带均为等波纹形式, 最适合设计片段常数特性的滤波器。其调用格式参看示例。

【例 9-44】设计一个低通滤波器, 其截止频率为 1600Hz, 阻带的起始频率为 2100Hz, 通带波纹的最大允许值为 0.02, 阻带波纹的最大允许值为 0.1, 采样频率为 8200Hz。

代码如下:

```
[n,fo,mo,w]=remezord([1600 2100],[1 0],[0.02 0.1],8200);
b=remez(n,fo,mo,w);
freqz(b);
```

或

```
[n,fo,mo,w]=remezord([1600 2100],[1 0],[0.02 0.1],8200,'cell');
b=remez(c{:});
freqz(b);
```

运行程序, 效果如图 9-41 所示。

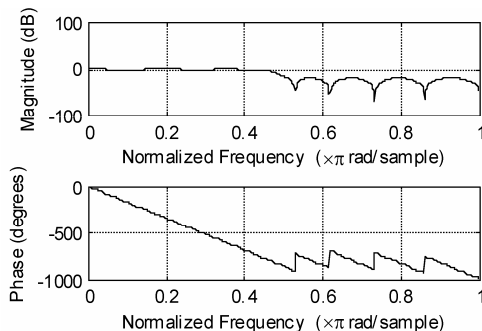


图 9-41 用 remez 设计的滤波器的频率响应

9.6 MATLAB实现功率谱估计

功率谱估计方法可以分成经典谱估计法与现代谱估计法。经典谱估计法又可分为直接法与间接法,直接法是利用快速傅里叶变换 FFT 算法对有限个样本数据进行傅里叶变换得到功率谱的方法,又称为周期图法;间接法是先得到样本数据的自相关函数估计,然后进行傅里叶变换得到功率谱的方法。现代谱估计主要是针对经典谱估计的分辨率低和方差性能不好等问题提出的。从现代谱估计的方法上,大致可分为参数模型谱估计和非参数模型谱估计。参数模型谱估计主要包括 AR 模型、MA 模型、ARMA 模型,以及最小方差谱估计等;非参数模型谱估计主要有基于矩阵特征值分解的功率谱估计,即特征向量谱估计与 MUSIC 方法谱估计。

这些功率谱的估计均可以通过 MATLAB 实现。以下给出利用函数 periodogram 实现直接法的功率谱估计。

【例 9-45】序列 $x(n) = \exp(j\omega_0 n - j\pi) + \exp(j\omega_1 n - 0.7j\pi) + e$ 为复正弦加白噪声的平稳信号,其中 $\omega_0 = 100\pi$, $\omega_1 = 50\pi$, e 为零均值的白噪声,信噪比 $S/N=100\text{dB}$ 。要求:(1)产生仿真数据;(2)利用直接法估计序列的功率谱。

该例的 MATLAB 代码如下:

```
clear;
Fs=1000; %采样频率
var=sqrt(1/exp(1.0));
n=0:1/Fs:1;
N=length(n);
e=var*randn(1,N);
w0=100*pi;
w1=50*pi;
xn=exp(j*w0*n-j*pi)+exp(j*w1*n-j*0.7*pi)+e;

%绘制信号波形
subplot(311);
plot(n,abs(xn));
xlabel('n');
title('x(n)=exp(j*w0*n-j*pi)+exp(j*w1*n-j*0.7*pi)+e')

%计算序列的 DTF
nfft=1024;
xk=fft(xn,nfft);
%计算序列的 PSD
pxx1=abs(xk).^2/N;
%绘制功率谱图形
index=0:round(nfft/2-1);
k=index*N/nfft;
plot_pxx1=10*log10(pxx1(index+1));
subplot(312);
plot(k,plot_pxx1);
ylabel('公式直接计算的功率谱');
```

```
%periodogram 函数计算的功率谱
window=boxcar(length(xn));
[pxx2,f]=periodogram(xn>window,nfft,Fs);
plot_pxx2=10*log10(pxx2(index+1));
subplot(313);
plot(k,plot_pxx2);
xlabel('periodogram 函数计算的功率谱');
```

运行程序，效果如图 9-42 所示。

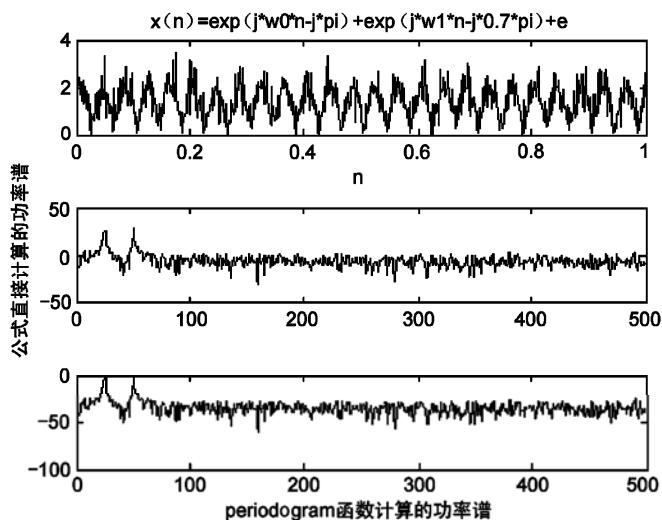


图 9-42 序列 $x(n)$ 的直接法功率谱估计曲线

从图 9-42 中可以看出利用公式直接计算的序列 $x(n)$ 的功率谱与利用函数 periodogram 计算得到的功率谱完全一致。

第 10 章 MATLAB外部程序接口应用

10.1 MATLAB数据接口

MATLAB 语言和其他程序设计语言一样，程序运行中的所有变量都保存在称为工作区的内存中，这些变量可以在程序中直接引用。但是工作区的大小是有限的，如果处理的数据较大，就需要和磁盘文件中的数据进行交换。有时要从外部设备中输入数据，有时要把程序处理过的数据输出到外部设备中。MATLAB 提供了多种不同层次的数据输入/输出函数。

10.1.1 通用文件I/O操作

MATLAB 支持某些特定格式和类型的数据文件（如图形文件、音频/视频文件、电子表格等）的读写操作，将该类文件中的数据导入到 MATLAB 的工作区，最简单的办法是使用数据导入向导（可通过选择 File 菜单项中的 Import data 命令或在命令窗口执行命令 `uiimport` 来激活它），而在 M 文件中则可以使用 MATLAB 输入/输出函数。

1. 文本数据的输入

在前面介绍过的 `load` 和 `save` 函数是 MATLAB 中为装载和存储数据提供的工具，但 `load` 函数只能读取以 ASCII 形式存储的，每一行数据为固定长度的文件。如果一个文件中的数据全部由 ASCII 字符组成，且数据间有间隔符（如空格、逗号、分号、制表位），则文件称为有格式文件。有格式文件可以使用文本输入函数读取数据，其调用格式为：

```
[A, B, C, ...]=textread(filename, format, N, param, value)
```

其中，A、B、C 是用于存放读取数据的向量，`filename` 为待操作的文件，`format` 用以控制读取的数据格式，由 % 加上格式符组成，常见的格式符如表 10-1 所示。N 指定重复使用该格式的次數，`param` 指定一些特殊操作，`value` 是与特殊操作有关的值，例如，跳过两行标题行可将 `'headerlines'` 参数设为 2。

表 10-1 数据格式描述符

格 式 符	含 义	格 式 符	含 义
%d	有符号的十进制整数	%e	指数形式的实数
%u	无符号的十进制整数	%f	小数形式的实数
%s	字符串	%c	字符

在 % 之后还可以加上数据宽度，例如 `%3d`，它控制读取的整型数据取 3 位数字；`%10.3f` 控制读取实型数据取 10 个字符（含小数点），小数部分占 3 位。

【例 10-1】假定文件 `textdemo.txt` 中有以下格式的数据：

Name	English	Chinese	Mathmatics
Wang	99	98	100
Li	85	77	91

Zhang	88	61	84
Zhao	78	52	100

此文件第一行为标题行，第二行至第五行的第一列为字符型，后三列为整型。从该文件中读取数据的命令如下：

```
[name, x, y, z]=textread('textdemo.txt', '%s %d %d %d', 4, 'headerlines', 1)
```

2. 多媒体文件的读写

MATLAB 除对文本数据进行操作外，还可以对图像文件、WAV 类型的音频文件和 AVI 类型的视频文件进行读写。下面介绍对音频文件和视频文件进行读写的函数。

(1) `info=mmfileinfo(filename)`: 查询音频或视频文件信息。

(2) `y=wavread(filename)`: 读取 WAV 格式的音频文件。

(3) `y=aviread(filename)`: 读取 AVI 格式的视频文件。

(4) `wavwrite(y, filename)`: 以 WAV 格式输出音频文件。

(5) `aviobj=avifile(y, filename)`: 将 MATLAB 的图像序列生成 AVI 格式的视频，建立 AVI 文件的句柄存放于 `aviobj` 中。

10.1.2 低级文件 I/O 操作

对于其他非通用格式文件的读写，可以采用 MATLAB 提供的低级文件 I/O 函数。因为这些函数是基于 ANSI 标准 C 语言库实现的，所以两者的格式和用法有许多相似之处。

1. 文件打开与关闭

对一个文件进行操作以前，必须先打开该文件，系统将为其分配一个输入/输出缓冲区。当文件操作结束后，还应关闭文件，及时释放缓冲区。

(1) `fopen` 函数

`fopen` 函数用于打开文件以供读写，调用格式参看示例。

(2) `fclose` 函数

`fclose` 函数用于关闭已打开的文件，调用格式参看示例。

2. 文本文件的读写

(1) `fscanf` 函数

`fscanf` 函数用于读取文本文件的内容，并按指定格式存入矩阵。调用格式参看联机帮助文档。

(2) `fprintf` 函数

`fprintf` 函数可以将数据按指定格式写入到文本文件中。调用格式参看示例。

【例 10-2】计算当 $x = [0.0, 0.1, 0.2, \dots, 1.0]$ 时， $f(x) = e^x$ 的值，并将结果写入文件 `demo1.txt`。代码如下：

```
x=0:0.1:1;
y=[x;exp(x)];
fid=fopen('demo1.txt','w');
fprintf(fid,'%6.2f %12.8f\n',y);
fclose(fid);
```

上述程序段中 `%6.2f` 控制 x 的值占 6 位，其中小数部分占 2 位。同样，`%12.8f` 控制指数函数 `exp(x)` 的输出格式。由于是文本文件，可以在 MATLAB 命令窗口用 `type` 命令显示其内容：


```
>> type demo1.txt
0.00    1.00000000
0.10    1.10517092
0.20    1.22140276
0.30    1.34985881
0.40    1.49182470
0.50    1.64872127
0.60    1.82211880
0.70    2.01375271
0.80    2.22554093
0.90    2.45960311
1.00    2.71828183
```

从此例可以看出，尽管 `fprintf` 的命令格式与 C 语言中的类似，但有一个主要的区别，即这里变量名只有一个 `y`，而输出的是 11 行数据，所以说 MATLAB 中的 `fprintf` 是矢量式的输出。

此外，可以利用 `fprintf` 函数实现数的类型转换。

【例 10-3】将十进制数转换为十六进制数。

代码如下：

```
>> a=[2 12 14 20 35];
fprintf('%9X\n',a+(a<0)*2^23)
```

运行代码输出为：

```
2
C
E
14
23
```

（3）`fgetl`与`fgets`函数

除上述对文本文件进行读写操作的函数外，读取文本文件的函数还有 `fgetl` 和 `fgets`，它们读取一行数据，当成字符串来处理。调用格式参看示例。

【例 10-4】读取例 10-2 生成的文件 `demo1.txt` 中的数据。

代码如下：

```
>> fid=fopen('demo1.txt','r');
while 1
    LI=fgetl(fid);
    if LI<0
        break;
    end;
    disp(LI);
end
fclose(fid);
```

该程序把文件 `demo1.txt` 一行一行地读入到 `LI` 中，每读一行在屏幕上显示一次，直至文件末尾。程序中结束 `while` 循环的方法是当读到文件末尾时，根据 `fgetl` 的规定返回 `LI=-1`，从而终止读操作。

程序的输出结果为：

```
0.00    1.00000000
0.10    1.10517092
.....
0.90    2.45960311
1.00    2.71828183
```

(4) textscan函数

从文件中导入数据，转换成一个指定格式的矩阵。格式如下：

```
W=textscan(fid, format, N, param, value)
```

其中，W 是用于存放读取数据的矩阵，fid 是待操作的文件标识，param 指定一些特殊操作，value 是与特殊操作有关的值。

3. 二进制文件的读写

(1) fread函数

从文件中读入二进制数据。其调用格式参看示例。

【例 10-5】假设文件 alpbet.txt 的内容是按顺序排列的 26 个大写英文字母，读取前 5 个字母的 ASCII 码和这 5 个字符。

代码如下：

```
fid=fopen('alpbet.txt','r');
c=fread(fid,5);
frewind(fid);          %将文件位置指针返回到文件的起始位置
d=fread(fid,5,'%char');
fclose(fid);
```

fscanf 与 fread 函数在读取数据时较灵活，不论数据文件中的数据是否具有确定的规律，均可以将数据文件的全部数据读入。而 load 函数在载入数据时，要求数据文件中的数据是有规律排列的，数据的排列类似矩阵或表格形式，否则不能成功读取数据。

(2) fwrite函数

fwrite 函数按照指定的数据类型将矩阵中的元素写入到文件中。其调用格式参看示例。

【例 10-6】建立一数据文件 magic5.dat，用于存放 5 阶魔方阵。

代码如下：

```
fid=fopen('magic5.dat','w');
ct=fwrite(fid,magic(5),'int32');
fclose(fid);
```

上述程序段将 5 阶魔方阵以 32 位整数格式写入文件 magic5.dat 中。同样，用户也可以读取文件 magic5.dat 的内容，下列程序完成对数据文件的读操作。

```
>> fid=fopen('magic5.dat','r');
[B,ct]=fread(fid,[5,inf],'int32')
fclose(fid);
```

运行代码输出为：

```
B =
    17    24     1     8    15
```

```

23    5    7    14   16
  4    6   13   20   22
 10   12   19   21    3
 11   18   25    2    9
ct =
25

```

4. 数据文件定位

当打开文件并进行数据的读写时，需要判断和控制文件的读写位置，例如判断文件数据是否已读完，或者需要读写指定位置上的数据等。MATLAB 自动创建一个文件位置指针来管理和维护读写文件数据的起始位置。

(1) fseek函数

用于定位文件位置指针，其调用格式参看示例。

(2) ftell函数

用来查询文件指针的当前位置，其调用格式参看示例。

【例 10-7】下列程序执行后，变量 four、position 和 three 的值是多少？

代码如下：

```

a=1:5;
fid=fopen('fdat.bin','w'); %以写方式打开文件 fdат.ат
fwrite(fid,a,'int16');      %将 a 的元素以双字节整型写入文件 fdат.ат
status=fclose(fid);
fid=fopen('fdат.ат','r'); %以读数据方式打开文件 fdат.ат
status=fseek(fid,6,'bof'); %将文件指针从开始位置向尾部移动 6 个字节
four=fread(fid,1,'int16') %读取第 4 个数据，并移动指针到下一个数据
position=ftell(fid)        %ftell 的返回值为 8
status=fseek(fid,-4,'cof'); %将文件指针从当前位置往前移动 4 个字节
three=fread(fid,1,'int16') %读取第 3 个数据
status=fclose(fid);

```

运行程序输出为：

```

four =      4
position =      8
three =      3

```

(3) feof函数

用来判断当前的文件位置指针是否到达文件尾部，调用格式为：

```
status=feof(fid)
```

测试结果返回 1 表示当前文件位置指针指向末尾，返回 0 表示没有指向末尾。

(4) ferror函数

用来查询最近一次输入或输出操作中的出错信息，其调用格式为：

```
[message, errnum]=ferror(fid)
```

其中 message 返回出错信息，errnum 返回错误编号。

10.1.3 MAT文件及其应用

MAT 文件是 MATLAB 数据存储的默认文件格式，以双精度二进制格式保存数据。MAT 文件为在不同平台或不同应用程序间移动数据提供了一种很方便的机制。

1. MAT文件

MAT 文件由 128 字节的 MAT 文件头和其后的数据单元组成。文件头包括 MATLAB 版本、数据和文件被创建的时间等信息。数据单元分为标志和数据两个部分，标志占 8 字节，包含数据类型、数据大小等信息。如果标志中的数据字节数小于 4，那么，MATLAB 使用压缩格式存储单元中的数据。

MATLAB 的 save 命令可以将 MATLAB 系统内部数据保存为 MAT 文件，而 load 命令可以将磁盘上的 MAT 文件中的数据读入到 MATLAB 系统中。此外，为了有效地管理 MAT 文件，以及在 MATLAB 外部读取和创建 MAT 文件，MATLAB 提供了一个子程序库，用户可以在 C/C++、FORTRAN 程序中直接调用这些子程序来创建和读取 MAT 文件。

MATLAB 提供的用于操作 MAT 文件的 API 函数封装于两个标准库文件中：libmat.lib 和 libmx.lib。前者用于对 MAT 文件的操作，后者用于对 MAT 文件中矩阵的操作。这两个库文件对于不同语言 and 不同版本的编辑器而有所不同，存放在 <MATLAB>\bin\win32 相应子文件夹中。此外，在 <MATLAB>\extern\include 中有与前面两个标准库文件对应的 .def 文件：libmat.def、libmx.def。其导出函数的原型位于同一目录下的 mat.h 和 matrix.h 中，mat.h 包含了 MAT 文件的创建、读写等函数的定义，matrix.h 包含了 MATLAB 中基本的数据类型、矩阵的定义和操作方法。

2. C语言MAT文件

常用 MAT 文件操作函数如下。

(1) 打开MAT文件

`MATFile *matOpen (const char *filename, const char *mode)`

其中 filename 为要操作的文件，mode 用来说明对文件的使用方式，可取以下值。

- r: 以只读方式打开文件。
- u: 以可读也可写的方式打开文件。
- w: 以只能写的方式打开文件。如果该文件中有内容，则删除原有内容。
- wz: 打开文件用于写入压缩数据。

(2) 关闭MAT文件

`int matClose (MATFile *mfp)`

其中 mfp 指向要操作的 MAT 文件，如果函数执行成功，返回 0，否则返回“EOF”。

(3) 向MAT文件中存入一个矩阵

`int matPutVariable(MATFile *mfp, const char *name, const mxArray *mp)`

此函数将一个 mp 指向的 mxArray 结构体写入 mfp 所指向的 MAT 文件中。如果文件中存在同名的 mxArray 结构体，那么将覆盖原来的值；如果不存在同名的 mxArray 结构体，则将此结构体添加到文件末尾。函数执行成功，返回 0，否则返回一个非零值。

(4) 向MAT文件中存放一个矩阵

`matPutArrayAsGlobal (MATFile *mfp, const mxArray *mp)`

执行此命令后，使用 load 命令装入这个 MAT 文件时，该矩阵对应的变量成为全局变量。

(5) 获取MAT文件中的变量列表

```
char *matGetDir (MATFile *mfp, const mxArray *mp)
```

函数执行成功, mfp 返回一个字符指针数组, 其中的每个元素指向 MAT 文件中的一个矩阵; 执行失败, mfp 返回一个空指针, num 为-1。如果 num=0, 则表示 MAT 文件中没有矩阵。

(6) 获取MAT文件的C语言FILE句柄

通过该句柄, 用户可以使用 C 语言的库函数 feof、ferror 来判断错误原因。

(7) 从MAT文件中读取一个矩阵

```
mxArray *matGetVariable (MATFile *mfp, const char *name)
```

如果函数执行成功, 在内存中创建一个命名为 name 的 mxArray 类型结构体对象, 并将读取的数据赋给该对象。

matGetDir、matGetFp、matGetVariable 函数通过 mxMalloc 函数分配内存, 在程序结束时, 必须使用 mxFree 函数释放内存。

(8) 从MAT文件中删除一个矩阵

```
int matDeleteVariable (MATFile *mfp, const char *name)
```

其中, name 为要删除的矩阵。如果函数执行成功, 将返回 0, 否则返回一个非零值。

3. 应用示例

【例 10-8】创建 MAT 文件。

```
#include<stdio.h>
#include<string.h> /*For strcmp()*/
#include<stdlib.h> /*For EXIT_FAILURE,EXIT_SUCCESS*/
#include"mat.h"
#define BUFSIZE 256
int main()
{
    MATFile *pmat; /*定义 MAT 文件指针*/
    mxArray *pa1,*pa2,*pa3;
    double data[9]={1.0 4.0 7.0 2.0 5.0 8.0 3.0 6.0 9.0};
    const char *file="mattest.mat";
    char str[BUFSIZE];
    int status;
    /* 打开一个 MAT 文件, 如果不存在, 则创建一个 MAT 文件, 如果打开失败, 则返回*/
    printf("Creating file %s...\n",file);
    pmat=matOpen(file,"w");
    if(pmat==NULL)
    {
        printf("Error creating file %s\n",file);
        printf("(Do you have write permission in this directory?)\n");
        return(EXIT_FAILURE);
    }
    /*创建三个 mxArray 结构体对象, 其中 pa1、pa2 分别为 3×3、2×2 的双精度实型矩阵*/
    pa1=mxCreateDoubleMatrix(3,3,mxREAL);
    if(pa1==NULL)
```

```

{
    printf("%s:Out of memory on line %d\n",_FILE_,_LINE_);
    printf("Unable to create mxArray.\n");
    return(EXIT_FAILURE);
}
pa2=mxCreateDoubleMatrix(3,3,mxREAL);
if(pa2==NULL)
{
    printf("%s:Out of memory on line%d\n",_FILE_,_LINE_);
    printf("Unable to create mxArray.\n");
    return(EXIT_FAILURE);
}
memcpy((void*)(mxGetPr(pa2)),(void*)data,sizeof(data));
pa3=mxCreateString("MATLAB:the language of technical computing");
if(pa3==NULL)
{
    printf("%s:Out of memory on line %d\n",_FILE_,_LINE_);
    printf("Unable to create string mxArray.\n");
    return(EXIT_FAILURE);
}
/*向 MAT 文件中写数据，失败则返回*/
status=matPutVariable(pmat,"LocalDouble",pa);
if(status!=0)
{
    printf("%s:Error using matPutVariable on line %d\n",_FILE_,_LINE_);
    return(EXIT_FAILURE);
}
status=matPutVariableAsGlobal(pmat,"GlobalDouble",pa2);
if(status!=0)
{
    printf("Error using matPutVariableAsGlobal\n");
    return(EXIT_FAILURE);
}
status=matPutVariable(pmat,"LocalString",pa3);
if(status!=0)
{
    printf("%s:Error using matPutVariable on line %d\n",_FILE_,_LINE_);
    return(EXIT_FAILURE);
}
/*清除矩阵*/
mxDestroyArray(pa1);
mxDestroyArray(pa2);
mxDestroyArray(pa3);
/*关闭 MAT 文件*/
if(matClose(pmat)!=0)
{
    printf("Error closing file %s\n",file);
}

```

```

        return(EXIT_FAILURE);
    }
    printf("Done\n");
    return(EXIT_SUCCESS);
}

```

4. FORTRAN语言MAT文件应用

(1) 常用MAT文件操作函数

常用 FORTRAN 语言 MAT 函数有以下几种。

Integer *4 function matOpen(filename,mode)

Integer *4 function matClose(mfp)

Integer *4 function matDeleteVariable(mfp,name)

Integer *4 function matGetDir(mfp,num)

Integer *4 function matGetVariable(mfp,name)

Integer *4 function matPutVariable(mfp,name,pm)

Integer *4 function matPutVariableAsGlobal(mfp, name, pm)

其中, mfp 为指向 MAT 文件的指针, name 是读取或写入 MAT 文件的矩阵名称, num 为 MAT 文件中矩阵的数目, pm 是 mxArray 结构体的指针, mode 为打开文件的方式。

(2) 应用示例

【例 10-9】读取例 10-8 产生的 MAT 文件。

```

integer matOpen,matGetDir,matGetVariable
integer mp,dir,adir(100),pa
integer mxGetM,mxGetN,matClose
integer ndir,i,stat
character *32 names(100)
mp=matOpen('matter.mat','r')
if(mp.eq.0) then
    write(6,*) 'Can"t open"mattest.mat".'
    stop
end if
!读取目录
dir=matgetdir(mp,dir)
if(dir.eq.0) then
    write(6,*) 'Can"t read directory.'
    stop
end if
call mxCopyPtrToPtrArray(dir,adir,ndir)
do i=1,ndir
    call mxCopyPtrToCharacter(adir(i),names(i),32)
end do
write(6,*) 'Directory of Mat-file:'
do i=1,ndir
    write(6,*) names(i)
end do
!读取矩阵
write(6,*) 'Getting array contents:'

```

```

do n=1,ndir
    pa=matGetVariable(mp,names(n))
    write(6,*)'Retrieved',names(n)
    write(6,*)'With size',mxGetM(pa),'-by-',mxGetN(pa)
    call mxDestroyArray(pa)
end do
stat=matClose(mp)
if(stat.ne.0) then
    write(6,*)'Error closing"mattest.mat".'
end if
end

```

10.2 MATLAB编译器的配置

MATLAB 编译器 (MATLAB Compiler, MCC) 是 MathWorks 公司为 MATLAB 开发环境提供的一种软件工具, 可将 M 语言编写的函数文件转换成 C、C++ 源代码, 这些源代码经过编译链接生成函数库、可执行程序、COM 组件等, 从而扩展 MATLAB 功能, 使 MATLAB 能够同其他高级语言, 例如 C/C++ 语言进行混合使用, 取长补短, 提高程序的运行效率, 丰富程序开发的手段。并且 M 文件是 ASCII 文件, 转换后成为二进制文件, 从而可以隐藏算法的源代码, 使用户可以保护开发版权。

未安装 MATLAB 编译器的用户如果想生成一个 MEX 文件, 就必须自己用 C 或 Fortran 语言来编写源代码, 这时用户必须了解 MATLAB 外部应用程序的接口 (External Interface, API) 和 MATLAB 如何表示它所支持的数据类型; 而安装了 MATLAB 编辑器, 就能利用 MATLAB 所提供的强大的数学、矩阵计算功能写出 M 文件, 然后用 MATLAB 编辑器直接编译为 C/C++ 源代码, 并在此基础上根据应用需要生成 MEX 文件、独立可执行应用程序、Excel 插值或 COM 对象等文件类型, 大大提高程序的运行速度, 提高代码的执行效率。这种方式创建的独立应用程序或者软件组件能够完全脱离 MATLAB 环境, 还可以与其他用户共享。

10.2.1 MATLAB编译器的配置

使用 MATLAB 编译器需要安装 MATLAB, 以及一种 MATLAB 支持的 C/C++ 语言编译器, 如 Visual C++ 6.0。为生成独立的应用程序, 还需要安装 MATLAB 相应的数字库。如果在程序中使用图形句柄, 则还需要安装相关图形库。

1. 编译器配置

应用 MATLAB 编译器前需要对编译器进行配置。编译器的配置文件是一个批处理文件, 它包括一些关于编译、预编译和链接阶段的编译器的设置信息。编译器配置步骤如下:

(1) 启动 MATLAB, 执行命令:

```
>> mbuild -setup
```

(2) 按提示选取一种编译器作为生成独立应用程序或共享库的默认编译器, 生成选项文件 compopts.bat, 并注册相关动态链接库文件。这样在以后使用 mcc 命令时, 系统将自动根据选项文件 compopts.bat 使用默认编译器。

mbuild 通过调用外部的 C/C++ 编译器, 把 M 文件翻译成 C/C++ 源码, 然后与 MATLAB 的 C/C++ 数字库、图形库链接, 得到可独立执行的可执行程序。

2. 验证mcc

将位于 MATLAB 文件夹下的子文件夹 extern\examples\compiler 中的示例文件 hello.m 复制到当前工作目录下，在 MATLAB 命令窗口执行命令：

```
>> mcc -m hello.m
```

将会生成 hello.exe。

10.2.2 编译指令

利用 MATLAB 提供的各种工具快速完成算法原型的开发、测试后，再通过 mcc 命令就可以将 M 文件转换为基于 C/C++ 语言数学函数库的 C/C++ 语言源代码文件。需要注意的是，mcc 命令只能转换 MATLAB 函数文件，而不能转换脚本文件，所以脚本类型的 M 文件应该先改写为函数类型的 M 文件。

1. mcc 命令的选项

mcc 是调用 MATLAB 编译器的命令，可以在 MATLAB 命令窗口使用，也可在 MS-DOS 下使用。命令格式为：

```
mcc [options] mfile1 [mfile2...mfileN]
```

mcc 命令选项及参数意义如表 10-2 所示。

表 10-2 mcc 命令选项及参数意义

选 项	意 义
-b	生成 Excel 公式中可用的函数
-d directory	指定生成文件的输出目录
-f filename	调用 mbuild 时使用指定选项文件
-g	产生调试信息
-l	生成 C 函数库
-m	生成独立应用程序，结果同-W main -T link:exe
-o outputfile	指定结果文件名
-B	指定生成的共享库类型及名称
	lib:<库文件名> C 类型库
	cpplib:<库文件名> C++类型库
-T target	指定输出类型。target 可以为以下值
	codegen 生成 C/C++包装文件
	compile:exe 编译为最终生成 EXE 类型文件的目标文件
	compile:lib 编译为最终生成 DLL 类型文件的目标文件
	link:exe 生成 EXE 类型文件
	link:lib 生成 DLL 类型文件
-W type	指定生成的包装文件类型。type 可以为以下值
	main 主程序
	lib:<库文件名> C 类型库
	cpplib:<库文件名> C++类型库
-?	显示帮助信息

2. mcc命令的使用

设 ademo.m 文件，内容如下：

```
function y=ademo()
x=-10:0.1:10;
y=cos(x);
plot(x,y,'+');
```

(1) 生成独立应用程序

生成独立应用程序采用命令：

```
>> mcc -m ademo.m 或 mcc -W main -T link:exe ademo.m
```

生成可执行文件 ademo.exe。运行该文件，结果如图 10-1 所示。

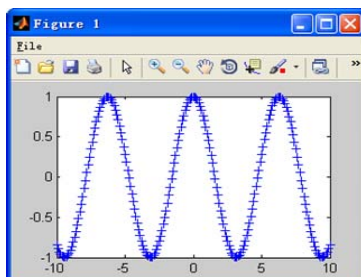


图 10-1 M 文件编辑后的结果

如果源文件不在当前工作目录，而产生的目标文件需要放置到/files/target 目录下，采用命令：

```
mcc -m -I/files/source -d/files/target ademo.m
```

(2) 生成共享库

- 生成 C 共享库采用命令：

```
mcc -I ademo.m 或 mcc -W lib:ademo -T link:lib ademo.m
```

该编译命令共创建 8 个文件，其中 ademo.lib 和 ademo.dll 是可以在 C/C++ 应用程序中调用的库文件，ademo.c、ademo.h 是 MEX 文件源代码及头文件，以便集成自定义的 C 代码文件。

- 生成 C++ 库采用命令：

```
mcc -W cpplib:ademo -T link:lib ademo.m
```

- 如果将 koo1.m、koo2.m 合并生成一个库文件 library_one.dll，采用以下命令：

```
mcc -W lib:library_one -T link:lib koo1 koo2
```

10.3 MATLAB引擎

10.3.1 MATLAB DDE服务器与引擎库

MATLAB 提供了客户-服务器 (Client/Server, C/S) 功能，利用动态数据交换服务功能和 ActiveX 自动化 (或 OLE 自动化) 服务功能，可以实现在其他程序中对 MATLAB 程序及函数的调用，并且通过 mxArray 与 MATLAB 工作空间交换数据，从而增强其他语言的数值计算能

力和数据的可视化能力。DDE 是在 Windows 环境下支持 C/S 计算模式的重要技术,两个应用程序之间可以通过交换数据来相互连接。在将 MATLAB 作为服务器访问时,必须提供服务器的名字、主题和项目。在具体的应用中,VC 的程序作为前端客户机,通过调用 MATLAB 引擎在后台与 MATLAB 服务器建立连接,实现动态通信。这种方法实现较为简单,不要求连接整个 MATLAB,只需要嵌入必要的 MATLAB 引擎库,可大大节省系统资源。而且利用网络,将引擎程序放在网络计算能力较强的机器上,可以使整个系统的运行速度加快。

MATLAB 引擎是 MathWorks 公司提供的一组函数库,它提供了一种在用户程序进程中与独立的 MATLAB 进程通信的方法,在 Windows 下使用 ActiveX 技术实现,使用组件对象模型(Component Object Model, COM)接口。MATLAB 的引擎函数库提供了在 C、Fortran 语言程序中打开和关闭引擎、与 MATLAB 工作空间交互数据、调用 MATLAB 命令等函数。

10.3.2 C语言MATLAB引擎

1. C语言MATLAB引擎函数

常用 C 语言引擎函数如下所示。

`engine *engOpen(const char *startcmd)`: 启动 MATLAB 引擎。

`int engClose(Engine *ep)`: 关闭 MATLAB 引擎。

`mxArray *engGetVariable(Engine *ep, const char *var_name)`: 从 MATLAB 引擎中获取一个 MATLAB 矩阵。

`int engPutVariable(Engine *ep, const char *var_name, const mxArray *array_ptr)`: 向 MATLAB 引擎发送一个 MATLAB 矩阵。

`int engEvalString(Engine *ep, const char *string)`: 执行一个 MATLAB 命令。

MATLAB 的子文件夹 `extern\include` 中的头文件 `engine.h` 包含了所有 C 语言引擎函数的定义。

2. C语言mx-函数

在 C 程序中调用 MATLAB 引擎函数的同时还用到 MATLAB 提供的接口函数中的 mx-函数,以完成对 mxArray 对象的操作。C 语言数学函数包含了 MATLAB 的矩阵运算核心,是以 mxArray 结构体(C++中是 `mwArray` 类)为核心构建的, mxArray 结构体定义在 MATLAB 的 `extern\include\matrix.h` 文件中。mx-函数有很多,具体的用法参看 MATLAB 帮助文件中 External Interfaces Reference 的 C MX-Functions。常用的 mx-函数如下所示。

`char *mxArrayToString(const mxArray *array_ptr)`: 将 mxArray 结构体转变为字符串。

`mxArray *mxCreateDoubleMatrix(int m, int n, mxComplexity ComplexFlag)`: 创建二维双精度类型 mxArray 矩阵。

`void mxDestroyArray(mxArray *array_ptr)`: 释放由 mxCreate 类函数分配的内存。

`int mxGetM(const mxArray *array_ptr)`: 获取矩阵的行数。

`int mxGetN(const mxArray *array_ptr)`: 获取矩阵的列数。

`void mxSetM(mxArray *array_ptr, int m)`: 设置矩阵的行数。

`void mxSetN(mxArray *array_ptr, int n)`: 设置矩阵的列数。

`double *mxGetPr(const mxArray *array_ptr)`: 获取矩阵实数部分的数据指针。

`double *mxGetPi(const mxArray *array_ptr)`: 获取矩阵虚数部分的数据指针。

`void mxSetPr(mxArray *array_ptr, double *pr)`: 设置矩阵实数部分的数据指针。

void mxSetPi(mxArray *array_ptr, double *pi): 设置矩阵虚数部分的数据指针。

void *mxCalloc(size_t n, size_t size): 在内存中分配 n 个大小为 size 字节的单元, 并初始化为 0。

3. C语言MATLAB计算引擎的编程

首先需要将 mxArray 转换成 MATLAB 中可操作的形式, 这可以分两步进行:

- 将 mxArray 转换成 MATLAB 可理解的形式。选择将一个自定义的数据结构复制到 mxArray 中。应注意 C 语言和 MATLAB 语言中数据存储方式的差别, 在 MATLAB 中矩阵是按列存储的, 而 C 语言数组元素是按行存储的。
- 将矩阵放入 MATLAB 计算引擎的工作区中, 可以用 engPutVariable 命令来完成。

【例 10-10】创建一个矩阵, 然后送到 MATLAB 计算引擎的工作区中, 绘制出结果图。

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include "engine.h"
#define BUFSIZE 256
int main()
{
    Engine *ep; /*定义 MATLAB 引擎变量*/
    mxArray *T=NULL;
    result=NULL;
    double time[10]={0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9};
    /*启动 MATLAB 计算引擎。如果在本地启动, 那么函数所带的参数字符串为空;*/
    /*如果在网络中启动, 则需要提供服务器名, 即 engOpen("服务器名")*/
    if(!(ep==engOpen("\0")))
    {
        fprintf(stderr, "\nCan't start MATLAB engine\n");
        return EXIT_FAILURE;
    }
    /*向新启动的 MATLAB 工作区放置数据*/
    T=mxCreateDoubleMatrix(1,10,mxREAL);
    memcpy((void*)mxGetPr(T),(void*)time,sizeof(time));
    engPutVariable(ep,"T",T);
    /*执行 MATLAB 命令*/
    engEvalString(ep,"D=0.5.*(-9.8).*T.^2;");
    engEvalString(ep,"plot(T,D);");
    engEvalString(ep,"time('Position vs.Time for a falling object');");
    engEvalString(ep,"xlabel('Time(seconds)');");
    engEvalString(ep,"ylabel('Position(meters)');");
    /*从 MATLAB 工作区获取计算结果*/
    result=engGetVariable(ep,"D");
    printf("The size of result is %d* %d\t\n",mxGetM(result),mxGetN(result));
    printf("Hit return to continue\n\n");
    fgetc(stdin);
    printf("Done!\n");
    /*释放岸上空间,关闭计算引擎*/
```

```

mxDestroyArray(T);
engEvalString(ep,"close");
engClose(ep);
return EXIT_SUCCESS;
}

```

4. C语言MATLAB计算引擎程序的编译

将源程序编写存盘后,使用 `mex` 命令对源程序文件进行编译。如果是第一次使用 `mex` 命令,系统将提示用户选取一种编译作为默认编译器。

```

>> mex
Select a compiler:
[1]Digital Visual Fortran version 6.0 in D:\ \Microsoft Visual Studio
[2] Lcc-win32 C 2.4.1 in D:\MATLAB~1\sys\lcc\bin
[3] Microsoft Visual C++ 6.0 in D:\Microsoft Visual Studio
[0] None

```

按照提示进行选择,设置完成后生成选项文件 `mexopts.bat`。这样在以后使用 `mex` 命令时,系统将会自动根据 `mexopts.bat` 使用指定编译器。

使用 `mex` 命令编译不同类型文件,必须使用参数 `-f` 指定合适的编译器选项文件,命令格式如下:

```
mex -f <MATLAB>\bin\win32\mexopts\optsfilename filename.c
```

使用 Visual C++ 6.0 编译器编译计算引擎程序和读写 MAT 文件的程序的选项文件为 `msvc60engmatopts.bat`。假设例 10-10 保存为 `v.c`,则编译该文件的方法是:

```

>> optsfile=[matlabroot '\bin\win32\mexopts\msvc60engmatopts.bat'];
>> mex('-f',optsfile,'v.c');

```

需要注意的是,此种方法编译的源文件中不能有中文(即使是注释)。编译此程序后得到一个可执行文件,运行结果如图 10-2 所示。

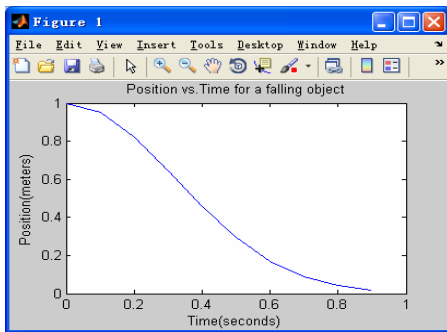


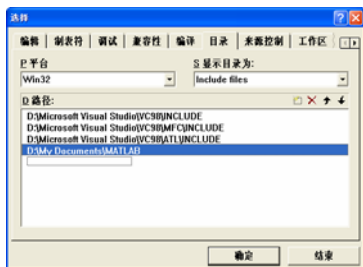
图 10-2 利用 C 计算引擎绘制的曲线

`mex` 是一个命令行的工具,用它来编译一个简单的文件是很合适的,但是当工程比较复杂时就需要在 Visual C++ 6.0 的集成环境中编译链接。步骤如下:

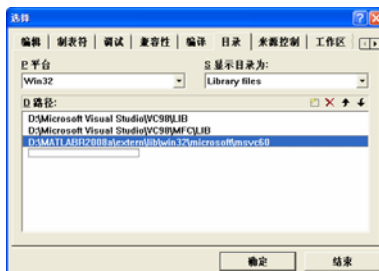
① 启动 Visual C++ 6.0,新建一个 Win32 Console Application 类型的项目(如果使用 Windows 资源,如对话框,则建立 Win32 Application 类型的项目),命名为 `cc`,从随后弹出的对话框中选择 An empty project 类型。

② 设置编译环境。单击【工具】菜单中的【选择】命令，弹出【选择】对话框，在弹出的【选择】对话框中选取【目录】选项卡，在【显示目录为】下拉列表框中选择【Include files】，在【路径】文本框中添加（假设 MATLAB 的路径为 D:\My Documents\MATLAB）D:\My Documents\MATLAB，然后再在【显示目录为】下拉列表框中选择【Library files】，在【路径】文本框中添加 D:\MATLABR2008\extern\lib\win32\microsoft\msvc60。

编译环境参数设置如图 10-3 所示。



(a) Include files 设置参数



(b) Library files 设置参数

图 10-3 设置编译环境

③ 设置项目连接参数。单击【工程】菜单下的【设置】选项，在弹出的【Project Settings】对话框中选择【Link】选项卡，在【对象/库模块】文本框中增加库 libmx.lib、libmat.lib 和 libeng.lib，如图 10-4 所示。



图 10-4 添加必要的库文件

④ 单击【工程】菜单中的【添加工程】选项中的【Files】命令，将编辑好的 v.c 文件加入到项目中。然后进行编译链接，如果无错误，将生成可执行文件 v.exe。

10.3.3 Fortran语言MATLAB引擎

1. Fortran语言MATLAB计算引擎函数

以下列出了常用的 Fortran 语言引擎函数及其功能。

integer *4 function engOpen(startcmd): 启动 MATLAB 引擎。

integer *4 function engClose(ep): 关闭 MATLAB 引擎。

integer *4 function engGetVariable(ep, name): 从 MATLAB 引擎中获取一个 MATLAB 矩阵。

integer *4 function engPutVariable(ep, name, pm): 向 MATLAB 引擎发送一个 MATLAB 矩阵。

integer *4 function engEvalString(ep, command): 执行一个 MATLAB 命令。

其中，startcmd 为引擎数（字符型），ep 为引擎指针（整型），name 为矩阵名（字符型），pm 为 mxArray 矩阵指针，command 为 MATLAB 命令（字符型）。

2. Fortran语言MATLAB计算引擎的编程

【例 10-11】在 Fortran 源程序中调用 MATLAB 引擎绘制多峰函数矩阵。

代码如下：

```
integer engEvalString,engClose,engOpen
integer ep,status
ep=engOpen('MATLAB')
if(ep==0) then
    write(6,*)'Can' 't start MATLAB engine'
    stop
endif
status=engEvalString(ep,'mesh(peaks);')
pause
if(engclose(ep)/=0) then
    write(6,*)'Can' 't close MATLAB engine'
endif
end
```

3. Fortran语言MATLAB计算引擎的编译

在 MATLAB 命令窗口使用 Visual Fortran 6.0 编译器编译计算引擎程序和读写 MAT 文件的程序的选项文件为 df60engmatopts.bat。假设例 10-11 保存为 eng.f90，则编译该文件的方法是：

```
optsfile=[matlabroot '\bin\win32\mexopts\df60engmatopts.bat'];
mex('-f',optsfile,'eng.f90');
```

在 Visual Fortran 6.0 集成环境中编译 Fortran 源程序的方法和步骤同 C 语言。如果编译链接无误，将生成可执行文件 eng.f90.exe，运行结果如图 10-5 所示。

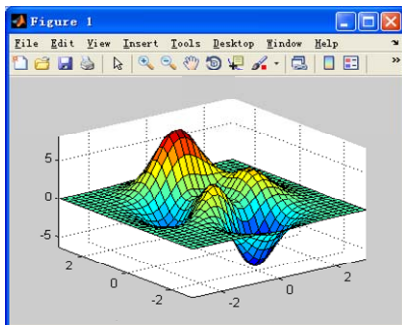


图 10-5 利用 Fortran 计算引擎绘制的曲线

10.4 Visual C++与MATLAB接口

MATLAB 开发环境提供了丰富的应用程序接口（API）库函数实现 C、C++与 MATLAB 的混合编程，通过 mex、mbuild、mcc 编译命令可以实现把 M 文件编译成 Visual C++环境下直接调用的 C/C++文件，或者把 C 文件编译成在 MATLAB 开发环境中直接调用的可执行文件。在 Visual C++中实现与 MATLAB 开发环境的接口主要有以下几种方法。

① 在 Visual C++中使用 MATLAB 引擎，调用应用程序接口（API）中的 Engine 函数库，调用 MATLAB 函数，但是该方法在编译成功后，会自动启动 MATLAB 开发环境。

② 在 Visual C++中直接调用 MATLAB 语言 C、C++数学函数 (Math Library), 通过 mex 和 mbuild 命令配置 MATLAB 语言 C、C++开发环境, 实现 Visual C++和 MATLAB 的接口, 编译形成独立于 MATLAB 环境的可执行.exe 文件。

③ 使用 C、C++语言编写 MEX 文件, 在 MATLAB 中通过 mex 命令编译成 MATLAB 可以直接运行的文件, 以实现 Visual C++与 MATLAB 的接口。

④ 使用 MathTools 公司开发的 MATCOM 可视化界面, 将 MATLAB 文件编译成 Visual C++可以直接执行的 C++文件, 生成的 C++文件可以同时兼顾 MATLAB 开发环境强大的数学运算功能和 C++语言运行速度快两个优点。使用 Visual MATCOM Add-in 插件, 在 Visual C++环境中直接编译运行 M 文件 (由于篇幅限制, 这种方法本书不介绍, 请读者参考相关资料)。

⑤ 使用 COM (Component Object Model) 组件可以实现 MATLAB 与 Visual C++的接口。通过 MATLAB COM Builder 可以把 MATLAB 开发的 M 文件、MEX 文件等做成组件, 使之成为独立的 COM 对象, 可以被其他支持 COM 语言的开发环境所使用, 如 Visual C++、Visual Basic、Java、Excel 等。

10.4.1 Visual C++调用MATLAB引擎

利用 MATLAB 应用程序接口 (API) 函数库的 Engine 函数库可以直接调用 MATLAB 开发环境函数, 一方面可以保持 C++代码的运行效率, 同时可以使用 MATLAB 丰富的数学函数库函数、各种工程领域工具箱函数等。但是使用 MATLAB 引擎技术, 生成的应用程序无法独立于 MATLAB 平台运行, 通过这种方式进行 MATLAB 函数调用, 必须要求系统安装 MATLAB 程序, 这给应用程序的移植和推广带来难度。

以下介绍的 Visual C++与 MATLAB 接口技术, 可以实现将 MATLAB 函数编译成.exe 文件、动态链接库文件 DLL 文件或编译成 COM 组件, 可以独立于 MATLAB 开发平台运行。

10.4.2 Visual C++使用数学函数库

MATLAB 开发环境的 C、C++数学函数库以 mxArray 结构体为数据核心, 使用 mcc 命令将 M 文件转换成的 C、C++文件, 可以直接在 Visual C++开发环境中使用, 但是编译后的 C、C++文件一般程序冗长、代码可读性较差。

在熟练掌握 C、C++数学函数库的基础上, 读者可以使用 mxArray 结构体自动编写基于应用程序接口 (API) 函数库 C、C++数学库的 C、C++文件, 可以极大地优化程序代码, 减少程序代码存储空间, 提高程序代码的执行效率和执行速度。

下面通过一个简单的示例来演示如何利用 mcc 命令创建独立于 MATLAB 平台运行的 exe 文件。

首先, 使用 MATLAB 的 M 语言建立用户所需要的特定功能 M 文件, 在这里建立一个实现魔方阵功能的 M 文件 magicCreate.m 文件。

```
function a=magicCreate(n)
a=magic(n)
```

然后, 从 %matlabroot%\extern\examples\compiler (%matlabroot% 为读者计算机系统中 MATLAB 的安装路径, 本机为 D:\MATLABR2008\extern\examples\compiler) 路径下复制 main_for_lib.c 和 main_for_lib.h 到当前的工作路径下, 以便 mcc 编译命令调用, 然后创建调用 magicCreate.m 文件的 C 语言, 程序如下:


```

#include <stdio.h>
#include <math.h>
#include "libPkg.h" /*编译建立的库头文件*/
main(int argc,char **argv)
{
    mxArray *N; /*输入变量矩阵指针*/
    mxArray *R=NULL; /*结果矩阵指针*/
    int n; /*默认的 M 文件输入变量数值*/
    /*获取命令行参数,如果命令行输入小于 2,则输入参数默认为 5*/
    if (argc>=2)
    {
        n=atoi(argv[1]);
    }
    else {
        n=5;
    }
    /*初始化 MCR 和 libPkg 函数库*/
    mclInitializeApplication(NULL,0);
    libPkgInitialize();
    /*得到输入参量的数值*/
    N=mxCreateDoubleScalar(n);
    /*调用 magicCreate.m 编译后的文件 mlfMagicCreate*/
    mlfMagicCreate(1,&R,N);
    /*释放内存空间*/
    mxDestroyArray(N);
    mxDestroyArray(R);
    /*结束 libRkg 库和 MCR*/
    libPkgTerminate();
    mclTerminateApplication();
}

```

其中, libPkg.h 为 mcc 命令编译后的库文件, 该文件结构清晰。

程序代码包含相关的头文件、libPkg.h 文件、主程序入口、变量的定义, 然后初始化 MCR 和 libPkg 函数库, 调用 magicCreate.m 编译后的文件 mlfMagicCreate, 调用结果用矩阵 R 存储, 最后释放内存空间, 并结束 libPkg 库和 MCR。

在程序结束之前及时地释放内存空间是一个很好的编程习惯。将该程序存储为 magicCreateC.c, 接下来使用 mcc 命令编译该文件, 在 MATLAB 命令行窗口中输入:

```
>> mcc -W lib:libPkg -T link:exe magicCreate magicCreateC.c main_for_lib.c
```

经过 mcc 命令编译后, 可以发现在当前路径下, 出现一个 magicCreate.exe 文件, 为一个可执行程序文件, 在 MATLAB 命令行窗口输入:

```

>> !magicCreate
Extracting CTF archive. This may take a few seconds, depending on the
size of your application. Please wait...
...CTF archive extraction complete.
a =

```

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

程序正常运行，由于命令行输入变量个数小于 2，因此 M 文件输入参数默认值为 5，程序产生一个 5 阶的魔方阵，如果需要输入第 2 个输入参数，则可以按照以下方式在 MATLAB 命令行窗口中输入：

```
>> !magicCreate 3 %创建一个 4 阶的魔方阵
a =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

10.4.3 Visual C++创建MAT文件

在 Visual C++6.0 开发环境中，同样可以利用 API 函数库的 MAT 函数库，写入和读出 MAT 数据文件。通过这种数据文件的读写，可以有效地实现 Visual C++6.0 环境同 MATLAB 开发环境的数据通信接口。以下 matCreatCDemo.c 文件为 Visual C++6.0 环境下实现 MAT 数据文件的读写操作，程序代码如下：

```
/*
 * matCreateCDemo.c
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "mat.h"
int main()
{
    //MATFile 文件定义，mxArray 变量定义
    const char *filename="MatDemo.mat";
    MATFile *file;
    int flag1,flag2,flag3;
    mxArray *pString, *pArray1,*pArray2;
    //初始数据定义
    double a1[]={0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0};
    double a2[]={9.0,4.0,5.4,6.4,3.2,1.5};
    //创建 MAT 文件
    printf("Creating file %s...\n\n",filename);
    file=matOpen(filename,"w");
    if(file==NULL)
    {
        printf("ERROR:Can not create file %s\n",filename);
        return(EXIT_FAILURE);
    }
}
```

```

}
//创建字符串 mxArray 数据结构变量
pString=mxCreateString("This is the MAT demo in C lanuage!Enojy it ~!");
if(pString==NULL)
{
    printf("ERROR:Unable to create string!");
    return(EXIT_FAILURE);
}
//创建矩阵 mxArray 数据结构变量
pArray1=mxCreateDoubleMatrix(3,3,mxREAL);
pArray2=mxCreateDoubleMatrix(2,3,mxREAL);
if((pArray1==NULL) || (pArray2==NULL))
{
    printf("Error:Cannot create the double matrix!");
    return(EXIT_FAILURE);
}
//将初始数据复制给 mxArray 数据结构变量
memcpy((void*)(mxGetPr(pArray1)),(void*)a1,sizeof(a1));
memcpy((void*)(mxGetPr(pArray2)),(void*)a2,sizeof(a2));
//向 MAT 文件中写入变量
flag1=matPutVariable(file,"variableString",pString);
flag2=matPutVariable(file,"variableDoubleMatrix1",pArray1);
flag3=matPutVariable(file,"variableDoubleMatrix2",pArray2);
if((flag1!=0) || (flag2!=0) || (flag3!=0))
{
    printf("Can not write variable into the file %s\n",filename);
    return(EXIT_FAILURE);
}
//释放内存空间
mxDestroyArray(pString);
mxDestroyArray(pArray1);
mxDestroyArray(pArray2);
//关闭 MAT 文件
if(matClose(file)!=0)
{
    printf("ERROR:Can not close %s file.\n",filename);
    return(EXIT_FAILURE);
}
//重新打开 MAT 文件
file=matOpen(filename,"r");
if(file==NULL)
{
    printf("ERROR:Can not open file %s\n",filename);
    return(EXIT_FAILURE);
}
//读取 MAT 文件的数据变量
pArray1=matGetVariable(file,"variableDoubleMatrix1");

```

```

pArray2=matGetVariable(file,"variableDoubleMatrix2");
pString=matGetVariable(file,"variableString");
if((pArray1==NULL) || (pArray2==NULL) || (pString==NULL))
{
    printf("ERROR:Can not reading variable from file %s\n",filename);
    return(EXIT_FAILURE);
}
//释放内存空间
mxDestroyArray(pArray1);
mxDestroyArray(pArray2);
mxDestroyArray(pString);
//关闭 MAT 文件
if(matClose(file)!=0)
{
    printf("ERROR:Can not close file %s\n",filename);
    return(EXIT_FAILURE);
}
printf("Complete!\n");
return(EXIT_SUCCESS);
}

```

在 Visual C++ 环境新建一个 Win32 Console Application 工程, 命名为 MatCreatCDemo, Visual C++ 环境的设置参考前面章节。输入以上程序段后, 运行结果如图 10-6 所示。

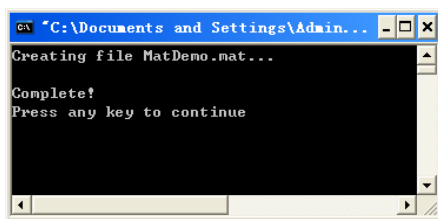


图 10-6 Visual C++ 环境下 MAT 文件运行结果

在 MatCreatCDemo 工程目录下, 可以看到新建的 MatDemo.mat 文件, 将工程目录下的 MatDemo.mat 文件复制到 MATLAB 工作路径下, 使用“load”指令可以查看 MatDemo 文件中的数据, 在 MATLAB 命令行窗口中输入:

```

>> load MatDemo
>> whos

```

Name	Size	Bytes	Class
variableDoubleMatrix1	3×3	72	double
variableDoubleMatrix1	2×3	48	double
variableString	1×43	86	double

Grand total is 58 elements using 206 bytes

10.4.4 应用COM实现Visual C++与MATLAB的接口

在 MATLAB 开发环境中使用 MATLAB COM Builder 工具可以将 M 文件和 MEX 编译为 COM 组件, 在 Visual C++ 开发环境中调用 COM 组件对象, 实现 Visual C++ 与 MATLAB 的接口。首先, 需要使用 mbuild 命令对编译环境进行配置。在 MATLAB 命令行窗口中输入:

```
>> mbuild -setup
Please choose your compiler for building standalone MATLAB applications:
Would you like mbuilder to locate installed compilers [y]/n? y
Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\MATLAB~1\sys\lcc
[2] Microsoft Visual C++ 6.0 in D:\Microsoft Visual Studio
[0] None

Compiler: 2
Please verify your choices:
Compiler: Microsoft Visual C++ 6.0
Location: D:\Microsoft Visual Studio

Are these correct [y]/n? y
Trying to update options file: C:\Documents and Settings\Administrator\Application
Data\MathWorks\MATLAB\R2008\compopts.bat
From template:
    D:\MATLAB~1\bin\win32\mbuildopts\msvc60comp.bat

Done . . .
DllRegisterServer in C:\Program Files\MATLAB71\bin\win32\mwcommmgr.dll succeeded
```

编译环境配置成功后, 就可以启动 MATLAB Builder 图形化接口界面, 在 MATLAB 命令行窗口中输入下面的指令, 即可启动 MATLAB Builder, 其操作界面如图 10-7 所示。

```
>> comtool
```

单击工具栏【File】菜单下的【New Project...】命令, 弹出如图 10-8 所示的新建工程窗口, 在【Component name】文本框中输入需要建立组件的名称“myComDemo”。注意组件名称避免和 M 文件或 MEX 文件名称相同。

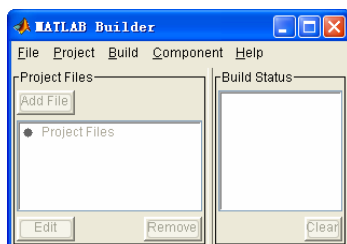


图 10-7 MATLAB Builder 启动界面

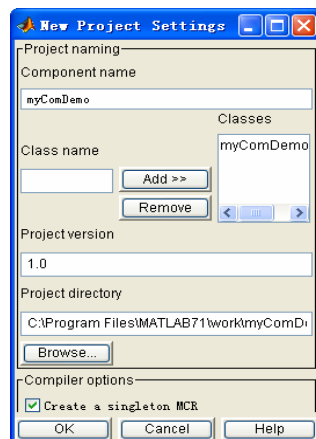


图 10-8 新建工程窗口

MATLAB COM Builder 会自动将 MyCOMDemoClass 加入到类列表中, 如图 10-9 所示。单击【OK】按钮, 弹出一个消息窗口询问当前 myComDemo 文件夹是否存在、是否创建, 单击【Yes】按钮就创建了一个 myComDemo 的组件。

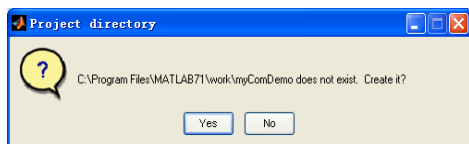


图 10-9 消息窗口

展开【myComDemoclass】前的“+”号，在 M-files 中添加所需要编译的 M 文件。单击工具栏【Project】菜单下的【Add files】命令，添加 M 文件到当前组件中，如图 10-10 所示。

添加完所需要的 M 文件或者 MEX 文件后，单击工具栏【Build】菜单下的【Com Object】命令，就可以将组件编译成在 C++ 代码中直接运行的 COM 组件。编译结果在右栏的信息中显示，如图 10-11 所示。

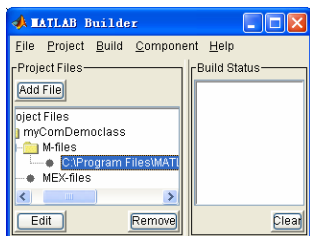


图 10-10 向 MATLAB Builder 中添加 M 文件或 MEX 文件

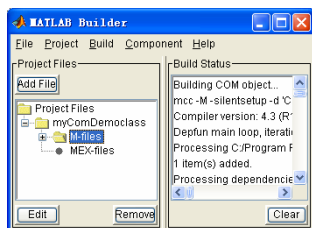


图 10-11 MATLAB Builder 编译成功的信息

在这里，有一点需要强调，如果读者计算机中 MATLAB 安装程序的安装路径为“D:\Program Files\MATLAB”，编译过程将无法通过，主要是因为 MATLAB 安装路径下出现了空格，编译时指定路径错误。按照以下提供的方法即可解决。

① 首先，在 MATLAB 命令行窗口中输入：

```
>> cd(prefdir)
>> copyfile('compopts.bat','compopts_old.bat') %子备份系统原文件，以防修改错误
>> edit compopts.bat
```

② 打开文件 compopts.bat 后，在程序代码的第 66 行，修改程序为：

```
set IDL_COMPILER=midl /nologo/win32 /I D:\Progra~1\MATLABR2008\extern\include
```

在这里假设读者的安装路径为“D:\Program Files\MATLAB”，如果不同，请读者自行修改。

③ 保存 compopts.bat 文件即可，此时再编译 COM 组件，就可以成功编译。为了使生成的 COM 组件可以在其他计算机上使用，需要对 COM 组件进行打包，单击工具栏【Component】菜单下的【Package Component】命令，MATLAB COM Builder 将会把发布该组件所需要的文件打包到一个和工程同名的可执行文件中，即 MS-DOS 批处理文件中，用来在其他计算机上注册该组件，如图 10-12 所示。

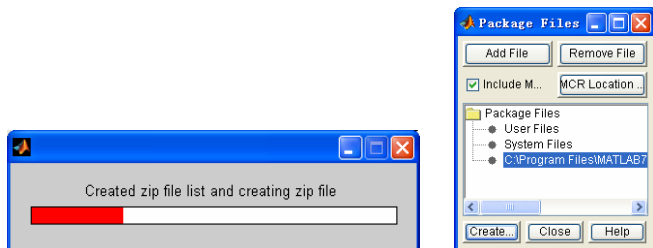


图 10-12 COM 对象进行打包发布

至此已经完成了 COM 组件的建立和发布，在 myComDemo\distrib 目录下，可以看到对外发布的 4 个文件。

_install.bat MS-DOS 批处理文件。

myComDemo.exe 自解压安装程序。

myComDemo.ctf 文件。

myComDemo_1_0.dll 动态链接库文件。

将 myComDemo\src 路径下的 myComDemo_idl.h 文件、myComDemo_idl.i.c 文件和 mwcomtypes.h 文件复制到 Visual C++工程目录下。在 Visual C++环境下，需要包含这 3 个文件。通过单击 Visual C++开发环境界面工具栏的【Tools】菜单下的【OLE/COM Object Viewer】命令，可以看到利用 MATLAB COM Builder 建立的 myComDemoclass 组件，如图 10-13 所示。

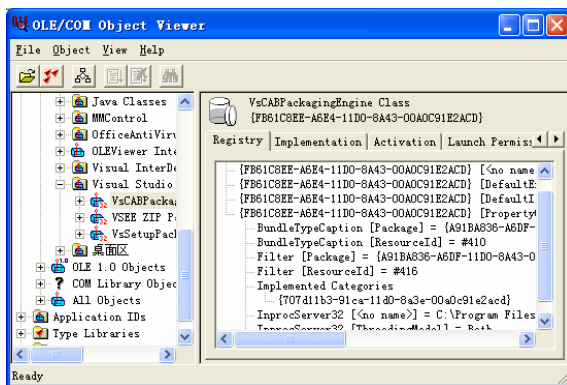


图 10-13 OLE/COM Viewer 组件信息

新建一个 Visual C++的 Win32 Console Application 工程，命名为 comDemo，采用所有默认属性，在 source files 文件夹下建立文件 com_file_example.cpp。C++程序可以使用早期绑定或后期绑定的方式来调用 COM 组件，一般多采用早期绑定调用 COM 组件的方式，对于一些特殊场合，也可能采用后期绑定的方式来调用 COM 组件。在 com_file_exaple.cpp 文件中输入以下程序段：

```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
//包含 COM 编译后产生的.h 文件和.c 文件
#include "myComDemo_idl.h"
#include "myComDemo_idl_i.c"
int main()
{
    //输入/输出变量的定义
    VARIANT x, y, out1;
    //COM 初始化
    HRESULT hr=CoInitialize(NULL);
    if(FAILED(hr))
    {
        printf("Can not initialize COM!\n");
    }
}
```

```

    return(EXIT_FAILURE);
}
//创建 COM 对象
ImyComDemoclass *pImyComDemoclass=NULL;
hr=CoCreateInstance
(CLSID_myComDemoclass,NULL,CLSCTX_INPROC_SERVER, IID_ImyComDemoclass,(void **)
&pImyComDemoclass);
if(FAILED(hr))
{
    printf("Can not create COM!\n");
    return(EXIT_FAILURE);
}
//设置输入/输出变量的类型和数值
x.vt=VT_R8;
x.dblVal=2.5;
y.vt=VT_R8;
y.dblVal=4.0;
//调用 COM 组件
hr=(pImyComDemoclass -> multiplyTwoValue(1, &out1, x, y));
//输出计算结果
cout<<"The input values were "<<x.dblVal<<" and "<<y.dblVal<<".\n\n";
cout<<"The multiply value of "<<x.dblVal<<" and "<<y.dblVal<<" is "<<out1.dblVal <<"\n\n";
//释放内存空间
if(pImyComDemoclass!=NULL)
{
    pImyComDemoclass -> Release();
    pImComDemoclass =NULL;
}
//注销 COM 组件
CoUninitialize();
return(EXIT_SUCCESS);
}

```

编译程序并运行，结果如图 10-14 所示。

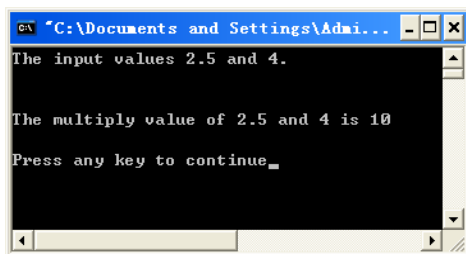


图 10-14 Visual C++中调用 COM 组件的运行结果

从 `com_file_example.cpp` 文件基本结构可以看出, 一般在 Visual C++ 中调用 MATLAB 生成的 COM 组件, 首先需要包含必要的头文件、变量的定义、COM 组件的初始化、COM 对象的创建、M 文件的调用、内存空间的释放、COM 组件的释放和注销。其中内存空间的释放是一个很好的编程习惯, 否则造成内存空间的泄露, 应用程序长时间运行后, 必须占满整个内存空间, 导致应用程序的崩溃。

以 `com_file_example.cpp` 文件为例, 在 Visual C++ 中调用 MATLAB 生成的 COM 组件可以按照以下几个步骤进行。

① 包含相应的头文件, 主要是 `_idl.h` 文件和 `idl_i.c` 文件, 如下所示:

```
# include "myComDemo_idl.h"
# include "myComDemo_idl_i.c"
```

② M 文件中相关输入/输出变量的定义, 使用 `VARIANT` 数据结构, 如下所示:

```
VARIANT x, y, out1;
```

③ 初始化 COM 组件, 如下所示:

```
HRESULT hr=CoInitialize(NULL);
if(FAILED (hr))
{
    printf("Can not initialize COM!\n");
    return(EXIT_FAILURE);
}
```

④ 创建 COM 对象, 如下所示:

```
ImyComDemoclass *pImyComDemoclass=NULL;
hr=CoCreateInstance
(CLSID_myComDemoclass,NULL,CLSCTX_INPROC_SERVER, IID_ImyComDemoclass,
(void **) &pImyComDemoclass);
if(FAILED(hr))
{
    printf("Can not create COM!\n");
    return(EXIT_FAILURE);
}
```

⑤ COM 对象初始化成功后, 就可以使用 COM 对象调用 M 文件, 如下所示:

```
hr=(pImyComDemoclass -> multiplyTwoValue(1, &out1, x, y));
```

⑥ 释放内存空间, 这是一个很好的编程习惯, 如下所示:

```
VariantClear (&x);
VariantClear (&Y);
VariantClear (&out1);
```

⑦ 释放 COM 组件, 如下所示:

```
if(pImyComDemoclass!=NULL)
{
```

```
pImyComDemoclass -> Release();  
pImComDemoclass =NULL;  
}
```

⑧ 注销 COM 组件，如下所示：

```
CoUninitialize()
```

第 11 章 MATLAB 在其他领域的应用

通过本书前面的学习,相信读者已经体会到 MATLAB 这个软件的确是一种优秀的科学计算语言,它不仅具有强大的编程能力,而且在各个应用领域中具有很多的应用价值。前面对 MATLAB 的介绍只起到抛砖引玉的作用,希望读者通过对本书的学习去挖掘 MATLAB 更多更大的价值。

下面再对 MATLAB 在其他一些领域中的应用进行简单介绍。

11.1 MATLAB 在电路中的应用

11.1.1 概述

1. 矩阵计算与线性电路分析

矩阵工具引入电路理论已有半个多世纪的历史。矩阵的引入使电路定律的表达更为精炼。由于把多变量的系统在形式上按单变量表示,整个理论显得更为简约,概念更为清晰,而且能从整体上掌握电路的状态。传统的克希霍夫定律、支路电流法、回路电流法以及节点电压法都可以以矩阵形式表示。

矩阵是 MATLAB 最基本的数据对象, MATLAB 的大部分运算或命令都是在矩阵运算的意义下执行的,而且 MATLAB 的矩阵运算定义在复数域上,这为电路分析带来了方便。

2. 微分方程求解与电路瞬态分析

当动态电路从某一稳定状态转换到另一稳定状态时,有些物理量(如 u_C 、 i_L 、 q 、 ψ)并不是突变的,而是需要一定的时间。在这期间,电路将呈现出和稳定状态不同的特别现象,这种现象为电路的过渡过程或瞬态现象。分析电路的瞬态现象时,可以建立关于电压和电流的微分方程,再按所给定的初始条件来进行求解。

MATLAB 提供了常微分方程初值问题的数值解法,利用有关函数可以进行电路瞬态分析。此外,还可以利用 MATLAB 符号计算或 Simulink 仿真来求解。

3. 图形功能与电路分析

利用 MATLAB 的图形功能可以绘制电路的各种响应曲线。

11.1.2 MATLAB 在电路中的应用示例

【例 11-1】三相不平衡交流电路分析。计算如图 11-1 所示的三相不平衡交流电路各支路电流($\dot{I}_a, \dot{I}_b, \dot{I}_c$),并绘制相量图。其中 $r_a = r_b = r_c = 5\Omega$, $r_{ab} = 6\Omega$, $r_{bc} = 10\Omega$, $r_{ca} = 15\Omega$, $E=220V$ 。

根据克希霍夫定律并代入数值, \dot{I}_{ab} 、 \dot{I}_{bc} 和 \dot{I}_{ca} 满足方程组:

$$\begin{cases} 16\dot{I}_{ab} - 5\dot{I}_{bc} - \dot{I}_{ca} = 220 \\ -5\dot{I}_{ab} + 20\dot{I}_{bc} - 5\dot{I}_{ca} = -110 - j110\sqrt{3} \\ -5\dot{I}_{ab} - 5\dot{I}_{bc} + 25\dot{I}_{ca} = -110 + j110\sqrt{3} \end{cases}$$

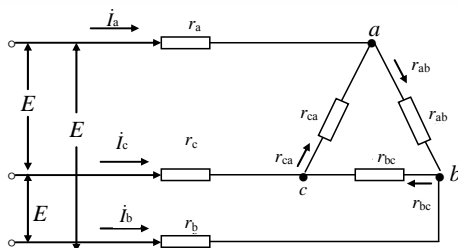


图 11-1 三相不平衡交流电路

用 MATLAB 先求 \dot{I}_{ab} 、 \dot{I}_{bc} 和 \dot{I}_{ca} 进而求各支路电流：

```
a=[16 -5 -5;-5 20 -5;-5 -5 25]; %系数矩阵
b=[220;-110-110*sqrt(3)*i;-110+110*sqrt(3)*i];
I=inv(a)*b; %解方程
%求各支路电流
Ia=I(1)-I(3)
Ib=I(2)-I(1)
Ic=I(3)-I(2)
h=compass([Ia,Ib,Ic]); %绘制相量图
set(h,'LineWidth',2);
```

运行代码，得到各支路电流为：

```
Ia =
    14.5783 - 6.5804i
Ib =
   -15.1084 - 7.4986i
Ic =
    0.5301 +14.0790i
```

各支路电流相量图，如图 11-2 所示。

【例 11-2】调谐振荡电路分析。分析如图 11-3 所示的调谐振荡电路 ($i_L = f(v) = \alpha + \beta v - \gamma v^3$, $\beta > 0, \gamma > 0$)，要求绘制振荡波形和相轨迹。

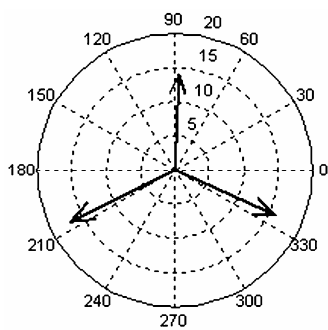


图 11-2 各支路电流相量图

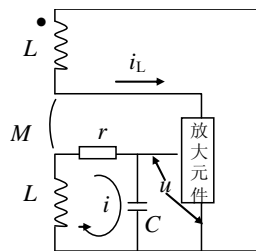


图 11-3 调谐振荡电路

根据克希霍夫定律，如图 11-3 所示调谐振荡电路的电流， i 满足下式（忽略流入放大元件的电流）

$$L \frac{di}{dt} + ri + \frac{1}{C} \int i dt = M \frac{di}{dt}$$

$$I = C \frac{dv}{dt}$$

由此两式可得到

$$LC \frac{d^2 v}{dt^2} + rC \frac{dv}{dt} + v = M \frac{di}{dt}$$

式中

$$i_L = f(v) = \alpha + \beta v - \gamma v^3, \beta > 0, \gamma > 0$$

再将上式写成

$$LC \frac{d^2 v}{dt^2} + (rC - M\beta + 3M\gamma v^2) \frac{dv}{d\tau} + v = 0$$

用 $t = \sqrt{LC}\tau$ 作为变换, 可得到

$$\frac{d^2 v}{d\tau^2} + \frac{1}{\sqrt{LC}}(rC - M\beta + 3M\gamma v^2) \frac{dv}{d\tau} + v = 0$$

如果 $rC - M\beta < 0$, 并设

$$\delta = \frac{3M\gamma}{\sqrt{LC}}, \quad \mu = \frac{M\beta - rC}{\sqrt{LC}}$$

则有

$$\frac{d^2 v}{dt^2} + (\delta v^2 - \mu) \frac{dv}{dt} + v = 0$$

设 $v = \sqrt{\frac{\mu}{\delta}} y$, 上式可写成

$$\frac{d^2 y}{dt^2} + \mu(y^2 - 1) \frac{dy}{dt} + y = 0$$

这是著名的范德波尔 (Van der Pol) 方程。它是一个非线性方程, 利用 MATLAB 可以求其数值解。设其初始条件为: $t = 0$, $y = a$, $\frac{dy}{dt} = b$ 。MATLAB 求常微分方程初值问题数值解的函数是对一阶常微分方程组设计的, 因此对高阶常微分方程, 需要先将它转化为一阶常微分方程组, 即状态方程。选择状态变量 $y_1 = \frac{dy}{dt}$, $y_2 = y$, 则可写出范德波尔方程的状态方程形式

$$\begin{cases} \frac{dy_1}{dt} = \mu(1 - y_2^2)y_1 - y_2 \\ \frac{dy_2}{dt} = y_1 \end{cases}$$

基于以上状态方程建立函数文件 xiupol.m:

```
function yd=xiupol(t,y)
yd(1)=0.1*(1-y(2)^2)*y(1)-y(2); % $\mu$  的值可以任意变化,此处取 0.1
yd(2)=y(1);
yd=yd'
```

求解微分方程, 并绘制振荡波形 (t, y) 和相轨迹 ($y, dy/dt$):

```
t0=0;tf=60; %确定积分区间
y0=[0;0.25]; %确定积分区间
[t,y]=ode45('xiupol',[t0,tf],y0); %求解微分方程
subplot(121);plot(t,y(:,2)); %绘制振荡方程
subplot(122);plot(y(:,2),y(:,1)); %绘制相轨迹
```

如图 11-4 所示是 μ 取 0.1 时的振荡波形和相轨迹, 从中可以清楚地了解起振时的波形变化。

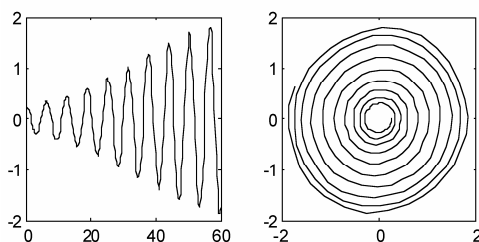


图 11-4 μ 取 0.1 时的振荡波形和相轨迹

当 μ 值很小时振荡波形和正弦波接近, 相轨迹极限环接近圆形。随着 μ 值的增大, 振荡波形将有显著失真, 相轨迹变形, 如图 11-5 所示。

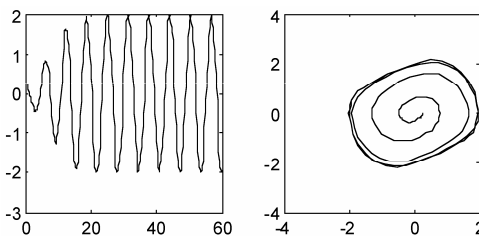


图 11-5 μ 取 0.4 时的振荡波形和相轨迹

MATLAB 定义在复数域上的矩阵计算、微分方程求解, 以及图形功能为电路分析创造了良好的条件。利用 MATLAB 来进行电路分析比传统方法更为简洁、高效。

11.2 MATLAB在图像处理中的应用

图像处理工具箱是 MATLAB 环境下开发出来的许多工具箱之一, 对应于 MATLAB R2008 工具箱版本为 V5.6, 它是以数字图像处理理论为基础, 用 MATLAB 语言构造得到的一系列用于图像数据显示与处理的 M 文件。图像处理工具箱可支持的图像处理的范围很广, 几乎包含了我们常见的所有图像处理函数, 从第一层次的图像变换 (包括空域变化)、图像运算、形态学运算, 第二层次的图像增强和滤波, 到第三层次的图像理解 (图像分析和感兴趣区域操作), 涵盖了绝大部分图像处理的内容。特别是, 它也包含了遥感图像中常用到的内容: 图像配准。

此外，对于图像的基本操作，比如读入、输出、保存等都得到了完美的支持。当然，我们也可以自己编写函数来扩展其功能。MATLAB 中的信号处理工具箱、神经网络工具箱、模糊逻辑工具箱和小波工具箱也用于协助执行图像处理任务。

11.2.1 图像变换

图像变换在数字图像处理与分析中起着很重要的作用，是一种常用的、有效的分析手段。图像变换的目的是使图像处理问题简化，更有利于图像特征提取，有助于从概念上加强对图像信息的理解。通过正交变换与酉变换改变图像的表示域及表示数据，为后续的处理工作带来极大的方便；常用的傅里叶变换使得图像的处理可以在频域中进行，使运算简便，图像经过变换后往往能反映出图像的灰度结构特征，便于分析；此外，许多变换可以使得能量集中在少数数据上，从而实现数据压缩，便于图像传输和存储。

图像变换的算法很多，包括线性变换、离散傅里叶变换、离散余弦变换、Hough 变换和沃尔什-哈达玛变换等。Hough 变换属于特征提取技术，在 MATLAB 中，实现这个功能的函数就是 `hough` 函数。检测的结果可以用 `houghline` 函数检测 H 中的叠加点，从而检测出 BW 的线段。

【例 11-3】用 `hough` 函数检测图像中的直线，Hough 变换的结果如图 11-6 所示。

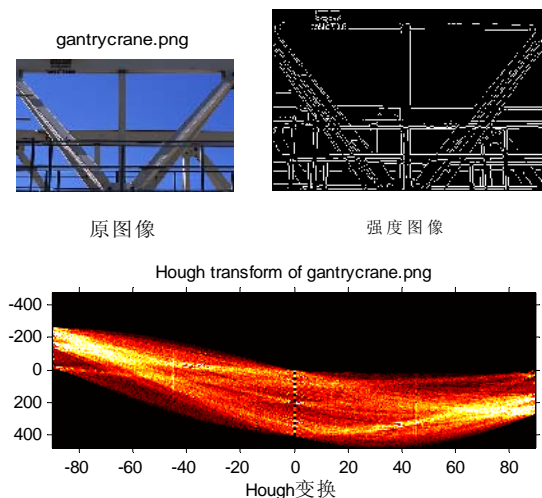


图 11-6 Hough 变换的结果

代码如下：

```
RGB=imread('gantrycrane.png');
I=rgb2gray(RGB); %转化成强度图像
BW=edge(I,'canny'); %提取边界
[H,T,R]=hough(BW,'RhoResolution',0.5,'ThetaResolution',0.5);
subplot(211);
imshow(RGB); %显示原图像
title('gantrycrane.png');
xlabel('原图像')
subplot(212);
imshow(imadjust(mat2gray(H)), 'XDData', T, 'YData', R, 'InitialMagnification', 'fit');
title('Hough transform of gantrycrane.png');
```

```

xlabel('Hough 变换');
axis on;axis normal;hold on;
colormap(hot);
P=houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
lines=houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure,imshow(BW);hold on;
max_len=0;
for k=1:length(lines)
    xy=[lines(k).point1;lines(k).point2];
    plot(xy(:,1),xy(:,2),'LineWidth',3,'Color','red');
    %显示线段的开头和结尾
    plot(xy(1,1),xy(1,2),'x','LineWidth',3,'Color','green');
    plot(xy(2,1),xy(2,2),'x','LineWidth',3,'Color','yellow');
    %检测最长线段的终点
    len=norm(lines(k).point1-lines(k).point2);
    if(len>max_len)
        max_len=len;
        xy_long=xy;
    end
end
title('强度图像')
%保存图像
imwrite(I,'xiu_1.bmp','bmp');
imwrite(I,'xiu_2.bmp','bmp');
H=H/(max(max(H)));
imwrite(I,'xiu_3.bmp','bmp');

```

H 矩阵上明显有高叠加点，也就是说，图像中包含有大量的直线。利用阈值取舍，我们可得到其检测的最终结果，如图 11-7 所示。

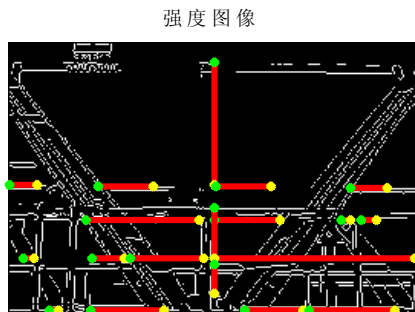
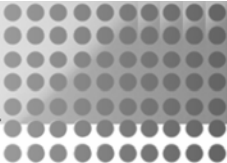


图 11-7 houghline 检测的最终结果

从图 11-7 可以看出，利用 Hough 可以有效地检测出图像的直线段。不过，为了得出较好的结果，我们必须多次调整检测的阈值。

11.2.2 MATLAB实现图像的边缘检测

边缘检测在图像处理与计算机视觉中占有特殊的位置，它是底层视觉处理中最重要的一环之一，也是实现基于边界的图像分割的基础。在图像中，边界表明一个特征区域的终结和另一个特征区域的开始，边界所分开区域的内部特征或属性是一致的，而不同区域内部的特征或属



性是不同的，边缘的检测正是利用物体和背景在某种图像特性上的差异来实现的。这种差异包括灰度、颜色或者纹理特征。边缘检测实际上就是检测图像特性发生变化的位置。

在 MATLAB 中，利用图像处理工具箱中的 `edge` 函数可以实现基于各种算子的检测边缘的功能，`edge` 函数调用格式如下：

```
[g,t]=edge(I, 'method', parameters)
```

其中，`I` 是输入图像，`method` 为表 11-1 列出的一种方法，`parameters` 是后面将要说明的另一个参数。在输出中，`g` 是一个逻辑数组，其值在 `I` 中检测到边缘的位置为 1，在其他位置为 0。参数 `t` 是可选的，它给出 `edge` 使用的阈值，以确定哪个梯度值足够大到可以称为边缘点。

表 11-1 函数 `edge` 中可用的边缘检测器

参 数		描 述
method	'roberts'	Robert 算子
	'sobel'	Sobel 算子
	'prewitt'	Prewitt 算子
	'log'	Log 算子
	'zerocross'	零交叉算子
	'canny'	Canny 算子

【例 11-4】以 Prewitt 算法进行边缘检测。

代码如下：

```
clear;
I=imread('rice.png');
imshow(I);xlabel('原始图像');
figure(2);imhist(I);           %显示图像的直方图
xlabel('图像的直方图');
I0=edge(I,'prewitt');          %自动选择阈值的 prewitt 算法
I1=edge(I,'prewitt',0.08);     %指定阈值为 0.08
I2=edge(I,'prewitt',0.05);     %指定阈值为 0.05
I3=edge(I,'prewitt',0.02);     %指定阈值为 0.02
figure(3);
subplot(221);imshow(I0)
xlabel('默认门限');
subplot(222);imshow(I1)
xlabel('门限为 0.08');
subplot(223);imshow(I2)
xlabel('门限为 0.05');
subplot(224);imshow(I3)
xlabel('门限为 0.2');
```

运行代码，效果如图 11-8 所示。

从例 11-4 可以看出，若要设定临界值，首先要看灰度直方图的分布，寻找其分界的地方为临界值，以本图像为例，取 0.08，其边缘效果就已经很明显了。



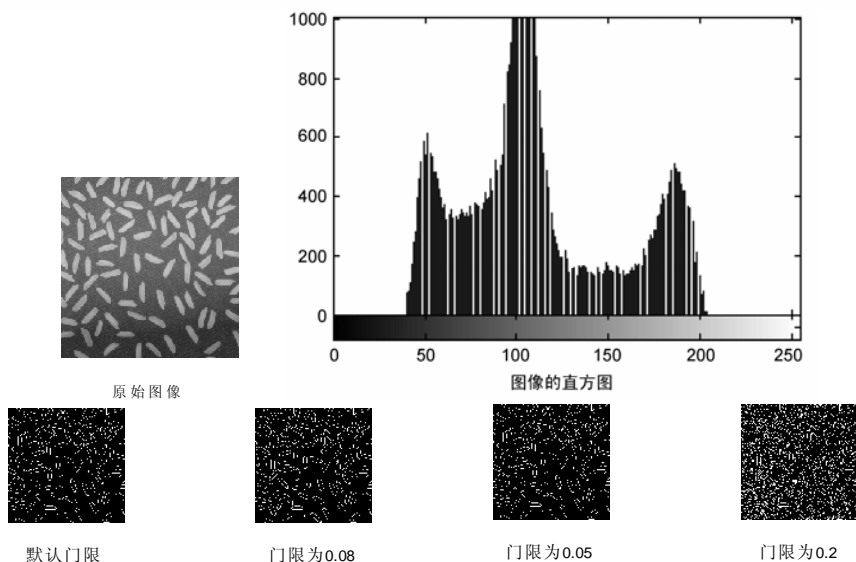


图 11-8 例 11-4 运行结果

11.2.3 MATLAB在遥感中实现图像直方图的匹配

遥感利用遥感器从空中来探测地面物体性质，它根据不同物体对波谱产生不同响应的原理，识别地面上的生物，并经记录、传送、分析和判读来识别地物。

遥感作为一门对地观测的综合性技术，它的出现和发展既是人们认识和探索自然界的客观需要，又具有其他技术手段与之无法比拟的特点。从字面上说，遥感就是从远处感觉事物，严格的定义是远远地去感觉某一特定对象的技术；而广义地讲，遥感是不直接接触地收集关于某一特定对象的某种或某些特定的信息，从而了解这个对象的性质。

遥感系统记录地球表面物质的反射和发射辐射通量。理想情况下，某种物质的特定波长会反射大量的能量，而另一种物质在同样的波长下反射的能量可能要小得多。这使得遥感系统记录的两种地物之间存在对比度。然而不同地物经常在可见光、近红外和中红外反射相似的辐射通量，使获取的影像对比度较低。另外，除了这些生物物理特征造成的明显低对比度外，人为因素也会对它产生影响。另一个导致遥感影像对比度低的因素是传感器的灵敏度。为了方便遥感影像分析人员对图像进行判读解译，需要对其进行增强处理。

在 MATLAB 中，使用 `histeq` 函数可以实现直方图匹配。

如图 11-9 (a) 所示显示了月亮的一幅图像 I，如图 11-9 (b) 所示显示了使用 `imhist(I)` 函数得到的直方图。由于这幅图像中一半区域较暗，所以直方图中的部分像素都集中在灰度级的暗端。乍一看，人们会认为利用直方图均衡化来增强图像是一种较好的方式，以便使较暗区域中的细节更加明显。然而，使用命令：

```
g=histeq(I,256);
```

得到如图 11-9 (c) 所示的结果表明，利用直方图均衡化方法在本例中并没有得到特别好的结果。对此，通过研究均衡化的图像如图 11-9 (d) 所示，可以看出原因。这里，我们看到灰度级已经移动到了灰度级的左半部分，因而输出图像出现了褪色现象。灰度级移动的原因是原始直方图中的暗色分量过于集中在 0 附近。从而，由该直方图得到的累计变换函数非常陡，因此才把在灰度级低端过于集中的像素映射到了灰度级的高端。所以程序如例 11-5 所示。

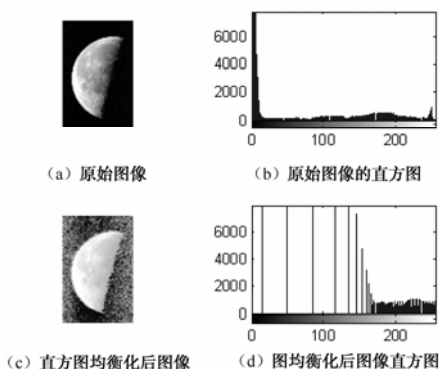


图 11-9 直方图均衡化效果

【例 11-5】直方图均衡化。

代码如下：

```
clear;
%显示原始图像
I=imread('moon.tif');
subplot(221);imshow(I);
title('(a)原始图像');
subplot(222);imhist(I);
title('(b)原始图像的直方图')
%对原始图像进行直方图均衡化
g=histeq(I,256);
subplot(223);imshow(g);
title('(c)直方图均衡化后图像');
subplot(224);imhist(g);
title('(d)图均衡化后图像直方图');
```

一种补偿这种现象的方法是使用直方图匹配。期望的直方图在灰度级低端应有较小的集中范围，并能够保留原图像直方图的大体形状。由图 11-9 (b) 可知，直方图主要有两个峰值，较大的峰值出现在原点处，较小的峰值出现在灰度级的高端。可使用多峰值高斯函数来模拟这种类型的直方图。

由于直方图均衡化在本例中出现的问题主要是原始图像 0 级灰度附近像素过于集中，因而较为合理的手段是修改该图像的直方图，使其不再有此性质。如图 11-10 所示显示了一个函数的图形（利用如下程序 xiuhist 得到，参数分别为 0.15, 0.05, 0.75, 0.05, 1, 0.07, 0.002），它不仅保留了原始直方图的大体形状，而且在图像的较暗区域中灰度级有较为平滑的过渡。

```
function p=xiuhist
repeats=true;
quitnow='x';
p=twomodegauss(0.15 0.05 0.75 0.05 1 0.07 0.002);
while repeats
    s=input('Enter m1,sig1,m2,sig2,a1,a2,k,OR x to quit:','s');
    if s==quitnow
        break
    end
    v=str2num(s);
    if numel(v)~=7
        disp('Incorrect number of inputs.');
```

```

        continue
    end
    p=twomodegauss(v(1),v(2),v(3),v(4),v(5),v(6),v(7));
    figure;plot(p);
    xlim([0 256]);
end

```

子函数 `p=twomodegauss(m1,sig1,m2,sig2,a1,a2,k)` 计算一个已经归一化到单位区域的双峰高斯函数，如图 11-10 所示，以便可以将它做成一个指定的直方图。

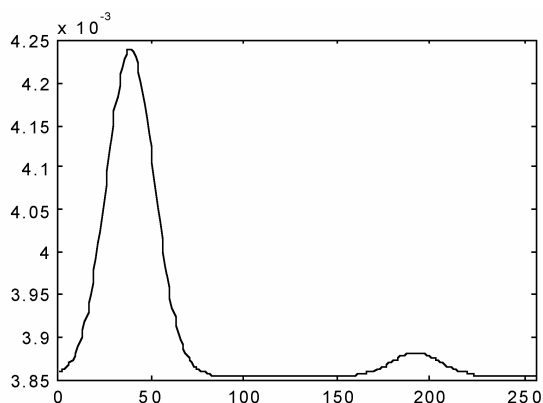


图 11-10 双峰高斯函数

```

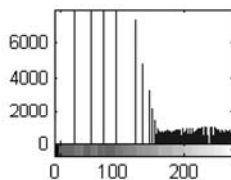
function p=twomodegauss(m1,sig1,m2,sig2,a1,a2,k)
c1=a1*(1/((2*pi)^0.5)*sig1);
k1=2*(sig1^2);
c2=a2*(1/((2*pi)^0.5)*sig2);
k2=2*(sig2^2);
z=linspace(0,1,256);
p=k+c1*exp(-((z-m1).^2)./k1)+c2*exp(-((z-m2).^2)./k2);
p=p./sum(p(:));

```

程序的输出 `p` 由该函数产生的 256 个等间隔点组成，它是我们所希望的指定直方图。利用下面的命令可以得到具有指定直方图的图像，如图 11-11 所示显示了最终结果。



(a) 直方图匹配增强结果图



(b) 期望图像直方图

图 11-11 直方图匹配增强

```
gg=histeq(I,p);
```

所用程序如例 11-6 所示。

【例 11-6】直方图匹配示例。

```

clear;
%获取一个指定的函数
p=xiuhist;
%使结果图像的直方图与获取函数图像一致

```

```
gg=histeq(I,p);
figure;
subplot(121);imshow(gg);
title('(a)直方图匹配增强结果图');
subplot(121);imhist(gg);
title('(b)期望图像直方图');
```

比较图 11-9 (b) 与图 11-11 (b) 可知,改进直方图后的结果增强是非常明显的。我们注意到,指定直方图是对原始直方图的恰当更改,而这正是在图像增强方面获得重大改进所要求的。

11.3 MATLAB在力学及工程结构中的应用

11.3.1 概述

在力学中涉及许多复杂的计算问题,例如非线性问题,对其求解析解有时是困难的, MATLAB 正是处理非线性问题的很好的工具,既能进行数值求解,又能绘制有关曲线,非常方便实用。

工程结构分析主要的根据是力学原理。经典力学原理基本上沿着两条路线进行。一条是基于牛顿运动定律。在静力分析中,主要遵循力的平衡原理,加上组成结构材料的本身关系和应变及位移的几何协调关系可以导出微分方程。另一条是基于功、能原理,它以能量原理(如最小势能原理、虚位移原理等)为基础,可以导出需要求解的积分方程。

不管是解微分方程还是解积分方程,均需要求出函数 $y = f(x)$, 使之满足方程并在边界上满足边界条件。对于简单的问题可以求得其解析解,但工程实际问题是复杂的,往往很难求得其实用的解析解,因此,应用计算机得到其数值解成了可行的解决问题的途径。常用的数值方法有差分法、有限元法、加权残值法、边界元法等,这些解法通常都有大量的矩阵运算,以及其他数值计算。MATLAB 具有强大的科学计算功能,这使得人们可以用它来代替 Fortran 等传统的编程语言。在计算要求相同的情况下,使用 MATLAB 编程,工作量会大大减少。例如,在结构动力学中利用 Fortran 求解结构的自由振动,需要调用 Jacobi 子程序求解矩阵的特征值及对应的特征向量,而且要求该矩阵必须为实对称矩阵,程序复杂,且对一般用户来说,要看懂程序算法实属不易,而采用 MATLAB 编制该自由振动的子程序时只需要调用两个函数:求逆矩阵的 inv 函数和求特征值和特征向量的 eig 函数。下面简单地介绍 MATLAB 在力学及工程结构分析中的应用。

11.3.2 MATLAB在力学及工程结构中的应用示例

【例 11-7】考虑空气阻力时抛射体质心的飞行轨迹问题。假设空气阻力的方向与速度向量相反,大小与速度的平方成正比。抛射体的受力图如图 11-12 所示。计算质点飞行的轨迹和距离。

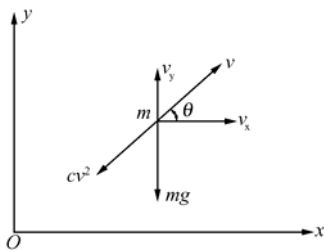


图 11-12 抛射体的受力图

质点运动方程为

$$\begin{cases} \frac{dx}{dt} = v_x \\ \frac{dy}{dt} = v_y \\ m \frac{dv_x}{dt} = -cv^2 \cos \theta = -cvv_x \\ m \frac{dv_y}{dt} = -cv^2 \sin \theta - mg = -cvv_y - mg \end{cases}$$

其中, c 为空气阻力系数。

一般要得到方程的解析解很困难,但可以用 MATLAB 来进行数值求解。设 $\mathbf{r} = [x, y, vx, vy]$, 则原方程等价于下面的一阶微分方程组

$$\frac{d\mathbf{r}}{dt} = \begin{bmatrix} r_3 \\ r_4 \\ -\frac{c}{m} \sqrt{r_3^2 + r_4^2} r_3 \\ -\frac{c}{m} \sqrt{r_3^2 + r_4^2} r_4 - g \end{bmatrix}$$

代码如下:

(1) 函数文件 cf.m

```
function rd=cf(t,r)
c=0.02;g=9.8;m=1;
vm=sqrt(r(3)^2+r(4)^2);
rd=[r(3);r(4);-c*vm*r(3)/m;-c*vm*r(4)/m-g];
```

(2) 主程序

```
clear;
y0=0;x0=0; %初始位置
v0=input('请输入初始输入速度(m/s):');
rho=input('请输入初始方向(度):');
tf=input('请输入飞行时间(s):');
vx0=v0*cos(rho*pi/180); %计算 x、y 方向的初始速度
vy0=v0*sin(rho*pi/180);
[t,r]=ode45('cf',[0,tf],[0;0;vx0;vy0]); %解微分方程
H=max(r(:,2)) %解微分方程
T=t(find(r(:,2)==H)) %求轨迹的最高点
L=min(r(find(r(:,2)<0),1)) %到最高点时间
plot([0,100],[0,0]);hold on; %计算射程
xlabel('x');ylabel('y');
plot(r(:,1),r(:,2)); %绘制运行轨迹
运行代码输出为:
请输入初始输入速度(m/s):67
请输入初始方向(度):38
请输入飞行时间(s):8.2
H =
```

27.3347
 $T =$
 1.8466
 $L =$
 81.8992

考虑空气阻力后抛射体的运动轨迹如图 11-13 所示。

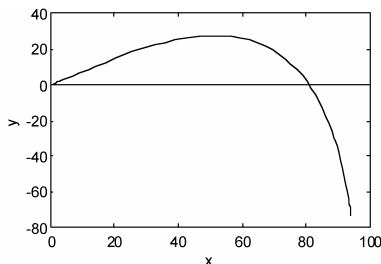


图 11-13 考虑空气阻力后抛射体的运动轨迹

【例 11-8】静不定问题，假设有一结构如图 11-14 (a) 所示，5 根直杆连接于节点 A、B、C、D、E 和 O。O 点受一个垂直向下的力 F 作用，各杆具有相同的弹性模量 E 和横截面积 A 。假设连接点允许旋转，所以杆上没有力矩作用。设 5 根杆的长度分别为 L_1 、 L_2 、 L_3 、 L_4 、 L_5 。这里， $L_1=L_5$ 且 $L_2=L_4$ 。5 根杆上受到的力分别为 F_1 、 F_2 、 F_3 、 F_4 、 F_5 ，如图 11-14 (b) 所示。由结构的对称性可知， $F_1=F_5$ ， $F_2=F_4$ 。所以总共有 3 个未知力 F_1 、 F_2 、 F_3 。设杆 L_1 和 L_3 所构成的夹角 $\angle AOC=\alpha_1$ ， L_2 和 L_3 所构成的夹角 $\angle BOC=\alpha_2$ 。同样 $\angle EOC=\alpha_1$ ， $\angle DOC=\alpha_2$ 。求 F_1 、 F_2 、 F_3 。

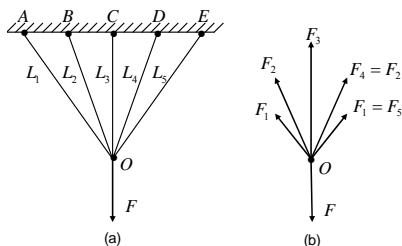


图 11-14 静不定杆及其受力图

节点 O 的静力平衡方程为

$$2F_1 \cos \alpha_1 + 2F_2 \cos \alpha_2 + F_3 = F$$

这里静力平衡方程有一个，但变量有 3 个，所以是静不定问题。为了求得方程的解，需要补充 2 个方程。

引入杆的伸长量作为第 4 个变量。由于弹性变形，可以认为在力 F_1 的作用下，杆 AO 的长度变为 $\Delta L_1 + L_1$ 。相似地，杆 BO 长度变为 $\Delta L_2 + L_2$ ，杆 CO 的长度变为 $\Delta L_3 + L_3$ ，根据胡克定律可以写出 3 个方程

$$\begin{aligned}\frac{\Delta L_1}{L_1} &= \frac{\sigma_1}{E} = \frac{F_1}{EA} \\ \frac{\Delta L_2}{L_2} &= \frac{\sigma_2}{E} = \frac{F_2}{EA} \\ \frac{\Delta L_3}{L_3} &= \frac{\sigma_3}{E} = \frac{F_3}{EA}\end{aligned}$$

其中, σ_1 是杆 AO 的截面上的应力。同理 σ_2 和 σ_3 是杆 BO 、杆 CO 上的应力。如果 $\Delta L_1 \ll L_3$, $\Delta L_2 \ll L_3$, 杆绕 A 、 B 点的旋转可忽略不计, 即当加上力 F 后, α_1 , α_2 的角度不变, 如图 11-15 所示。

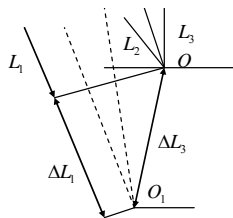


图 11-15 节点 O 产生的位移

在 F 的作用下, O 点运动到 O_1 点, 所以可得到

$$\Delta L_1 = \Delta L_3 \cos \alpha_1$$

$$\Delta L_2 = \Delta L_3 \cos \alpha_2$$

代入到上面的 3 个方程, 可得

$$\frac{L_1}{EA} F_1 - \Delta L_3 \cos \alpha_1 = 0$$

$$\frac{L_2}{EA} F_2 - \Delta L_3 \cos \alpha_2 = 0$$

$$\frac{L_3}{EA} F_3 - \Delta L_3 = 0$$

再联立静力平衡方程, 并用矩阵形式表示为

$$\begin{bmatrix} 2 \cos \alpha_1 & 2 \cos \alpha_2 & 1 & 0 \\ \frac{L_1}{EA} & 0 & 0 & -\cos \alpha_1 \\ 0 & \frac{L_2}{EA} & 0 & -\cos \alpha_2 \\ 0 & 0 & \frac{L_3}{EA} & -1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \Delta L_3 \end{bmatrix} = \begin{bmatrix} F \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

设长度 $L_3=1000$ mm, 力 $F=25\ 000$ N, 弹性模量 $E=205\ 000$ N/mm², 横截面面积 $A=100$ mm²。设 $AB=BC$, $AC=L_3/2$, 程序如下:

```
alpha1=atan(1/2); alpha2=atan(1/4);
L3=1000;F=25000;E=205000;A=100;
L1=L3/cos(alpha1);L2=L3/cos(alpha2);
C=[2*cos(alpha1),2*cos(alpha2),1,0;L3/(E*A),0,0,-cos(alpha1);0,L2/(E*A),0,-cos(alpha2);0,0,L3/(E*A),-1];
B=[F;0;0;0];
X=C\B
X =
    1.0e+003 *
    5.2200
    5.0642
    5.8362
    0.0003
```


很容易验证这个解满足 O 点的受力平衡方程:

```
>> Y=2*X(1)*cos(alpha1)+2*X(2)*cos(alpha2)+X(3)
Y =
    25000
```

【例 11-9】简支梁受力图如图 11-16 所示, 求其弯矩、转角和挠度。已知 $L=8\text{m}$, $q=500\text{N/m}$, $F_0=1000\text{N}$, $M_0=800\text{N}\cdot\text{m}$, $E=200\times 10^9\text{N/m}^2$, $I=2\times 10^6\text{m}^4$ 。

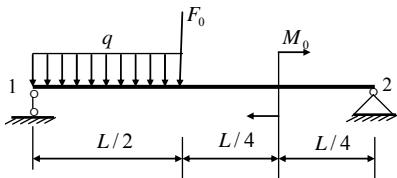


图 11-16 简支梁受力图

从材料力学的知识可知, 由弯矩求转角要经过一次不定积分, 而由转角求挠度又要经过一次不定积分, 通常这是很麻烦而且容易出错的, 而在 MATLAB 中, 可用 `cumsum` 函数或 `cumtrapz` 函数进行近似的不定积分, 只要 x 取得足够密, 其结果将相当准确, 且程序非常简单。

由平衡方程可求出支撑反力 N_1 和 N_2

$$N_1 = \left(\frac{qL}{2} \cdot \frac{3L}{4} + F_0 \cdot \frac{L}{2} - M_0 \right) / L, \quad N_2 = \frac{qL}{2} + F_0 - N_1$$

故各段弯矩方程为

$$\begin{aligned} M_1 &= N_1 x = q \cdot \frac{x^2}{2}, & 0 \leq x \leq \frac{L}{2} \\ M_2 &= N_2(L-x) - M_0, & \frac{L}{2} \leq x \leq \frac{3L}{4} \\ M_3 &= N_2(L-x), & \frac{3L}{4} \leq x \leq L \end{aligned}$$

对 M/EI 进行积分, 得转角 A , 再进行一次积分, 得到挠度 Y , 每次积分都要出现一个待定积分常数

$$A = \int_0^x \frac{M}{EI} dx + C_1 = A_0(x) + C_1$$

此处设 $A_0(x) = \text{cumtrapz}(M) * dx / EI$ 。

$$Y = \int_0^x A dx + C_2 = \int_0^x A_0(x) dx + C_1 x + C_2 = Y_0(x) + C_1 x + C_2$$

此处设 $Y_0(x) = \text{cumtrapz}(A_0) * dx$

两个待定积分常数 C_1 和 C_2 可由边界条件 $Y(0)=0$ 及 $Y(L)=0$ 确定

$$\begin{aligned} Y(0) &= Y_0(0) + C_2 = 0 \\ Y(L) &= Y_0(L) + C_1 L + C_2 = 0 \end{aligned}$$

于是可得

$$\begin{bmatrix} 0 & 1 \\ L & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} -Y_0(0) \\ -Y_0(l) \end{bmatrix}$$

由此编写 MATLAB 代码如下:

```
L=8;q=500;F0=1000;
M0=800;E=200e9;I=2e-6;
N1=(3*q*L^2/8+F0*L/2-M0)/L;
N2=(q*L^2/8+F0*L/2+M0)/L;
x=linspace(0,L,101);
dx=L/100;
M1=N1*x(1:51)-q*x(1:51).^2/2; %分3段用数组列出M的表达式
M2=N2*(L-x(52:76))-M0;
M3=N2*(L-x(77:101));
M=[M1,M2,M3];
A0=cumtrapz(M)*dx/(E*I);
Y0=cumtrapz(A0)*dx; %由M积分求转角(未计积分常数)
C=[0,1;L,1]/[-Y0(1),-Y0(101)]; %由转角积分求挠度(未计积分常数)
A=A0+C(1); %由边界条件求积分常数
Y=Y0+C(1)*x+C(2); %求转角和挠度的完整值
subplot(311);plot(x,M);grid on;
title('弯矩曲线');
subplot(312);plot(x,A);grid on;
title('转角曲线');
subplot(313);plot(x,Y);grid on;
title('挠度曲线');
```

运行代码, 效果如图 11-17 所示。

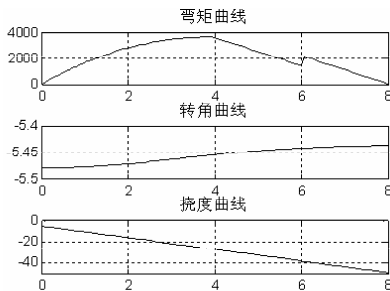


图 11-17 弯矩、转角和挠度曲线

梯形积分累加函数 `cumtrapz` 与梯形积分函数 `trapz` 的不同在于它逐点给出积分值, 因而得出一个积分序列, 而 `trapz` 只给出积分到终点的一个值。

11.4 MATLAB在优化设计中的应用

11.4.1 概述

优化设计是用数学规划理论和计算机自动探优技术来求解最优化问题。对工程问题进行优化设计, 首先需要将工程设计问题转化成数学模型, 即用优化设计的数学表达式描述工程设计

问题。然后,按照数学模型的特点选择合适的优化方法和计算程序,运用计算机求解,以获得最优设计方案。优化方法和计算程序的优劣,主要从可靠性和有效性两个方面进行考虑。解题成功率高、能够适应多变量和各种函数形态的数学模型的优化方法,计算程序的可靠性高。收敛速度快、通用性强和前置准备工作简便(包括对算法的流程图和参数符号的熟悉和定义,目标函数和约束函数子程序的编写,初始条件的给定、搜索过程及信息的输出与分析等内容)的优化方法和计算程序的有效性好。

11.4.2 MATLAB在优化设计中的应用示例

【例 11-10】无约束优化问题。已知梯形截面管道如图 11-18 所示,参数是底边长度是 c ,高度是 h ,斜边与底边的夹角是 θ ,横截面面积是 $A=64516 \text{ mm}^2$ 。管道内液体的流速与管道截面的周长 s 的倒数成比例关系。试按照使液体流速最大确定该管道的参数。

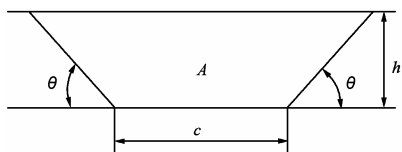


图 11-18 梯形截面管道

管道截面的周长

$$s = c + \frac{2h}{\sin \theta}$$

由管道横截面面积

$$A = ch + h^2 \cot \theta = 64516$$

得到底边长度的关系式(与 h 和 θ 相关)

$$c = \frac{64516 - h^2 \cot \theta}{h} = \frac{64516}{h} - h \cot \theta$$

将上式代入管道横截面周长的计算式中,得到

$$s = \frac{64516}{h} - h \cot \theta + \frac{2h}{\sin \theta} = \frac{64516}{h} - \frac{h}{\tan \theta} + \frac{2h}{\sin \theta}$$

因此,取与管道截面周长有关的独立参数 h 和 θ 作为设计变量,即

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} h \\ \theta \end{bmatrix}$$

为使液体流速最大,取管道截面周长最小作为目标函数,即

$$\min f(X) = \frac{64516}{x_1} - \frac{x_1}{\tan x_2} + \frac{2x_1}{\sin x_2}$$

这是一个二维无约束非线性优化问题。

(1) 建立目标函数文件xiumin.m

```
function f=xiumin(x)
a=64516;
f=a/x(1)-x(1)/tan(x(2)*pi/180)+2*x(1)/sin(x(2)*pi/180);
```

(2) 调用fminsearch函数求解

```
>> x0=[25;45];
[x,fval]=fminsearch('xiumin',x0)
```

运行代码得到优化结果为：

```
x =
    192.9982
     60.0000
fval =
    668.5656
```

即梯形截面高度 $h=192.9982$ mm，梯形截面斜边与底边夹角 $\theta=60^\circ$ ，梯形截面周长 $s=668.5656$ mm。

【例 11-11】证券投资组合问题。设金融市场上有两种风险证券 A 和 B，它们的期望收益率分别为 $r_A=12\%$ ， $r_B=18\%$ ，方差分别为 $\sigma_A^2=10$ ， $\sigma_B^2=1$ ， $\sigma_{AB}=0$ 。同时市场上还有一种无风险证券，其收益率为 $r_f=6\%$ ，设计一种投资是组合方案，使得风险最小。

投资者把资金投放于有价证券以期获得一定收益的行为就是证券投资。它的主要形式是股票投资和债券投资，证券投资的目的是价值增值。这是证券投资的收益特性，通常可用收益率指标表示证券的收益特性。证券预期收益率的不确定性使证券投资具有风险特性。具有投资风险的证券称为风险证券。无风险证券投资是指把资金投放于收益确定的债券，如购买国库券。若无风险投资的收益率为 r_f ，则 r_f 是常数。一般而言，风险证券投资往往有超过 r_f 的预期收益率，风险证券的预期收益率越高，其投资风险也越大。为了避免或分散投资风险，获取较高的预期收益率，证券投资可以按不同的投资比例对无风险投资和多种风险证券进行有机组合，即所谓证券投资组合。

对一个证券组合，用 $R=(r_1, r_2, \dots, r_n)^T$ 表示这 n 种证券的收益率， σ_{ij} 表示证券 i 和证券 j 的收益率之间的协方差， $i, j=1, 2, \dots, n$ ， $X=(x_1, x_2, \dots, x_n)^T$ 表示证券组合的投资权重，若同时投资于无风险证券，并设其收益率为 r_f ，则投资决策模型即为

$$\begin{aligned} \min \sigma_p^2 &= \sum_{i=1}^n \sum_{j=1}^n x_i \sigma_{ij} x_j = X^T W X \\ \text{s.t.} &\begin{cases} \sum_{i=1}^n x_i r_i + \left(1 - \sum_{i=1}^n x_i\right) r_f = x_p \\ \sum_{i=1}^n x_i = 1 \end{cases} \end{aligned}$$

其中， σ_p^2 为投资组合收益的方差，代表投资组合的风险， x_p 表示投资组合的期望收益率， W 为协方差矩阵。

设分别以比例 x_1 购买股票 A, 比例 x_2 购买股票 B, 比例 x_3 购买无风险债券, 则可建立单目标规划问题

$$\begin{aligned} \min \sigma_p^2 &= X^T W X \\ \text{s.t.} \quad &\begin{cases} x_1 r_A + x_2 r_B + x_3 r_f = r_p \\ x_1 + x_2 + x_3 = 1, \quad x_1 \geq 0, x_2 \geq 0 \end{cases} \end{aligned}$$

假定期望收益率 $r_p = 10\%$, 解决最优问题的程序如下:

(1) 函数文件xf.m

```
function f=xf(x)
v=zeros(3,3);
v(1,1)=10;
v(2,2)=1;
f=x'*v*x;
```

(2) 主程序代码

```
format rat
ra=0.12;
rb=0.08;
rf=0.06;
rp=0.1;
x0=[1,1,1]'/3;
Aeq=[ra,rb,rf,1,1,1];
beq=[rp,1]';
Lb=[0,0,-100]';
options=optimset('LargeScale','off','Display','off');
x=fmincon('xf',x0,[],[],Aeq,beq,Lb,[],[],options)
format short
```

运行结果为:

```
x =
    6/19
   20/19
   -7/19
```

结果表明, 为了获得 10% 的期望收益率, 应以无风险利率从银行贷款 7/19 单位, 将贷款和手中已有的单位现金的总和投资股票, 其中的 6/19 购买 A 股票, 20/19 购买 B 股票。

参 考 文 献

- [1] 徐昕, 李涛等. MATLAB 工具箱应用指南——控制工程篇. 北京: 电子工业出版社, 2000.
- [2] 楼顺天等. 基于 MATLAB 的系统分析与设计: 模糊系统. 西安: 西安电子科技大学出版社, 2001.
- [3] 韩力群. 人工神经网络理论设计与应用. 北京: 化学工业出版社, 2002.
- [4] 魏海坤. 神经网络结构设计的原理与方法. 北京: 中国铁道出版社, 2000.
- [5] 李士勇. 模糊控制、神经控制和智能控制论. 哈尔滨: 哈尔滨工业大学出版社, 1996.
- [6] 张瑞丰. 精通 MATLAB 6.5[M]. 北京: 中国水利水电出版社, 2004.
- [7] 苏金明, 王永利. MATLAB 7.0 实用指南上册. 北京: 电子工业出版社, 2004.
- [8] 张威. MATLAB 外部接口编程. 西安: 西安电子科技大学出版社, 2004.
- [9] 飞思科技产品研发中心. MATLAB 6.5 辅助优化计算与设计. 北京: 电子工业出版社, 2003.
- [10] 王世香. 精通 MATLAB 接口与编程. 北京: 电子工业出版社, 2006.
- [11] 赵震宇, 徐用懋著. 模块理论和神经网络的基础和应用. 北京: 清华大学出版社, 1996.
- [12] 王丹力, 赵剡, 邱治平. MATLAB 控制系统 设计 仿真 应用. 北京: 中国电力出版社, 2007.
- [13] 何强, 何英. MATLAB 扩展编程. 北京: 清华大学出版社, 2002.
- [14] 王华, 李有军, 刘建存. MATLAB 电子仿真与应用教程. 北京: 国防工业出版社, 2006.
- [15] 赵振宇, 徐用懋. 模糊理论和神经网络的基础与应用. 北京: 清华大学出版社, 1997.
- [16] 李士勇. 模糊控制、神经控制和智能控制论. 哈尔滨: 哈尔滨工业大学出版社, 1996.
- [17] 王素立, 高洁, 孙新德. MATLAB 混合编程与工程应用. 北京: 清华大学出版社, 2008.
- [18] 唐向宏, 岳恒立, 郑雪峰. MATLAB 及在电子信息类课程中的应用. 北京: 电子工业出版社, 2006.
- [19] 董振海. 精通 MATLAB 7 编程与数据库应用. 北京: 电子工业出版社, 2007.
- [20] 王正林, 王胜开, 陈国顺. MATLAB/Simulink 与控制系统仿真. 北京: 电子工业出版社, 2005.
- [21] 龙脉工作室, 刘会灯, 朱飞. MATLAB 编程基础与典型应用. 北京: 人民邮电出版社, 2008.